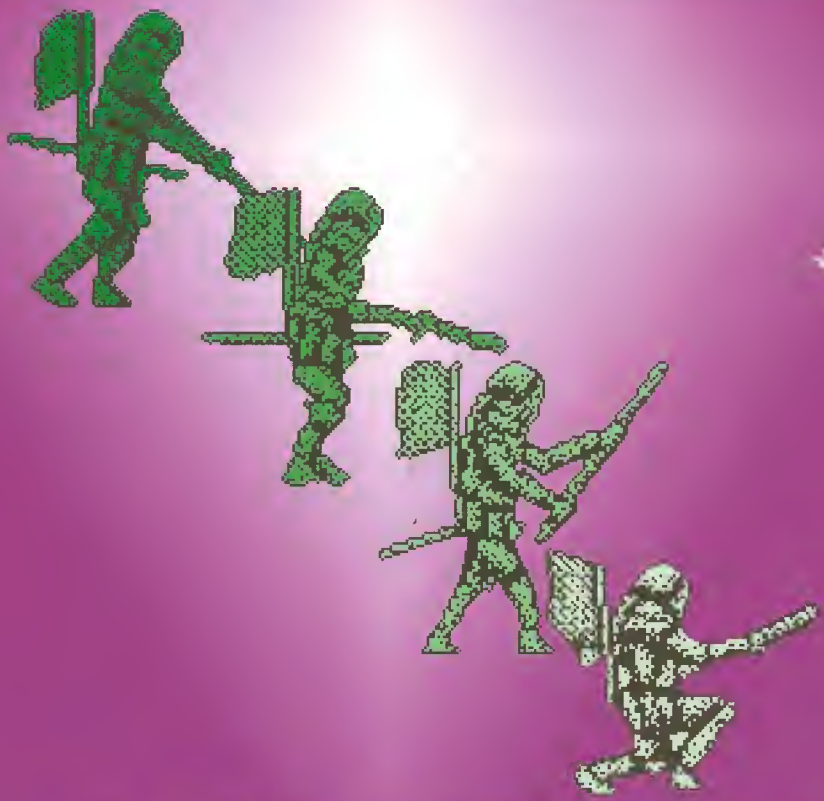
Writing Role Playing Games in AMOS



Dicon Peeke

Writing Role Playing Games in AMOS

by Dicon Peeke

Writing Role Playing Games in AMOS

ISBN 07457 0247 3

Published by:

Kuma Books Ltd
12 Horseshoe Park
Pangbourne
Berks
RG8 7JW

Tel 0734 844335
Fax 0734 844339

Contents

Chapter 1

Planning the Campaign.

- What's needed to get going.
- A very short history of the Amiga and what it can do.
- D&D adventures through the ages.
- The Lure of the dungeon and the pull of combat.
- Laying out your ideas on paper.
- Memory constraints and hard drives versus floppy.
- Mapping on a grid and building walls in the dungeon.
- Tome Junior explained.

People have been enjoying role playing games for many hundreds of years. In the very beginning, after a battle perhaps, the captains would discuss battle tactics with the help of pebbles and stones. Some would take the enemy's part and mock battles would be fought on the "If he had done this then you could have done that" method. Later, in the Regency Period, retired generals would set up mock battles using a team of worthies to act as adjudicators to set out the terrain, weather and number of lead soldiers.

In America, the first real role-playing game was devised by a team of people headed by Garry Gygax. Called Dungeons and Dragons, it was played with paper and pencil, look-up tables and dice. All the character classes, hit points and magic systems had their origins in this game and most have been used in computer versions of role-playing games since.

The computer role-playing games that we know today have an interesting ancestry. In the late seventies, a computer game called Rogue started to appear on university computer systems. The graphics were character based and the gameplay was very simple, but in essence it laid the foundations for the computer role-playing games of today.

One of the most famous of these was created on a great mainframe by Crowther and Woods in America as a diversion for the programmers between number-crunching projects. These games were played only by a privileged few, but, like most good secrets,

the idea of an adventure or role-playing game in which the computer played the games-master leaked out. In those early heady days, the great American corporations lost thousands of dollars in expensive computer-time as dragons and trolls roamed where once only dollars and cents had flowed. These adventures were simple and without graphics; there were no mice and, unbelievably, no screen. All responses were printed out.

The Amiga arrived in Britain and immediately became the top selling home computer. With Ham Mode, up to 64 variable colours in half bright and a built-in floppy disk drive (with options for hard drive and an extra floppy disk drive) it became a firm favourite almost overnight. Thousands have been sold and have revolutionised the personal computer and its availability.

AMOS Basic from Europress, written by Francois Lionet, arrived shortly afterwards and proved to be the best Basic Language for the Amiga in the world. With the many AMOS commands and easy-to-understand editor, the design and publishing of 'home-grown' programs began to spread. Some games from the big software houses are Compiled AMOS programs. There are still a lot of good AMOS programs being created and played that will stand the test of time.

The idea of exploring strange environments and fighting strange creatures has been with man since he walked on the face of the Earth. Stories of strange and fantastic creatures were the stock-in-trade of every wandering peddler and mountebank. Tony Crowther had the genius of combining the ancient teller of tales with the very modern computer to bring about a story that we could react to, and more importantly, would itself react to whoever played it. Now for the first time in history, when you started a story that you knew, you entered a changing world where things would talk back or fight. There was also the strong possibility of the simulated demise of the player, over and over again without harm.

With the improved graphics capabilities on Amiga, you can see people and unreal creatures advancing towards you. Using the mouse you can point and click on areas of the screen and pick up objects and initiate any action you choose. With the Amiga's advanced four channel sound chip you can have atmospheric music, sound effects and even hear the approach of the nasties. More importantly, all these actions are controlled by a very fast, intelligent machine that can arrange some surprising combinations of actions as the story demands. The Amiga can also Multi-task. Not even big Blue (PC) can do real multi-tasking.

If we are to design an exciting graphic adventure, we must first try to visualise the game with its characters and surroundings. Is it damp, gloomy and set in a dungeon or in the streets of a long abandoned desert town, uncovered from the sands by a sand-storm? Decide if there is a roof to the area, or if you intend to explore under the stars. Are you the only person in this new world or do you direct a team of treasure crazed adventurers and try to keep them out of trouble?

In this book, the game we are to create has a party of four brave explorers exploring a dungeon. All the examples in this book and on the Library disk can be changed or modified, but for now, we will keep the layout small and simple. In this game, characters usually move sixteen pixels for each 'turn'. I'll explain more in detail in a later chapter. To conserve precious memory, each part of the picture is made up of small blocks of graphics measuring sixteen by sixteen pixels, called tiles. These can be used over and over again like pieces of a jigsaw puzzle that combine and re-combine to make a picture.

To design your dungeon, just for now, start with paper, pen and coloured pencils. Grid paper is available from most stationers, but at this early stage a small sheet of paper with a ruled grid will do. In the game, the images of the walls and floor are stored as numbers, so that the computer knows where to 'past' the pictures for walls and floor and therefore, where the characters can go. To start with, draw a grid sixteen boxes by sixteen boxes. This will give you an area of two hundred and fifty six boxes.

There have to be walls all round the edge of the dungeon to stop your team falling over the edge of this virtual world; so in each box on the edge of the grid, pencil in a zero. Add passages and rooms as you like, putting in "#" for walls and a " " (space) for where your character or monster may move. As you can see from the diagram, it's a little difficult to follow, so that's where the coloured pencils come in useful. Shade in the paths very lightly and the structure of the dungeon will be much clearer. The other reason for using pencils is that alterations are easier to make. When you are happy with the result, use a pen to ink in the ones and zeros. This should look roughly like diagram [1]. Don't worry if it is not exactly right as you start; it's only a rough guide to layout and design.

```
################
#          #
############## #
### ###    ## #
##    ###### #
#######      #
### ### ###   #
###   ###   #
######       #
####### ########
#          #
### ########## #
#        ## #
# ########### #
##          #
################
```

diagram [1]

As your party of intrepid explorers wanders around in the dark, dripping depths of your dungeon, they are to meet many challenges. The simplest are doors that will slow their passage and give the player a puzzling time trying to open them. These can be drawn in as the number two and the squares coloured-in with a different colour in order to highlight them. In places all the 'spaces' cluster together to show an open area. Put in a number nine. This is to be an area pointer for the monster or fighter chosen by the computer when the game demo is run. The result of the additions should look like the diagram [2]. You can if you like put a small picture of a monster in.

```
################
#          #
############## #
### ###    ## #
##    ###### #
#######      #
### ### ###   #
###   ###   #
######       #
#######2########
#          #
### ########## #
#       9 ## #
# ########### #
##          #
################
```

diagram [2]

As you can see from diagram [2], mapping a large dungeon on paper can get somewhat tedious and difficult to follow. So, to make designing the game easier for you, on the disk is a dungeon mapper called Tome Junior designed by Aaron Fothergill. This utility will only work on Amos Classic. The program is Public Domain so it can be used freely in your program. This will allow you to design a huge dungeon and test it out before adding it to the game. You can load it by inserting the Library disk in drive "A" and loading the program "TOME.AMOS" into the Amiga.

I will have to mention The Tome Map Editor Series 4, the big brother of Tome Junior, again, designed by Aaron Fothergill. This piece of software is a must for all games creators. It speeds up the design of maps and has more features and help routines than you can shake an Ork at. There are map tile animators and a special language called Maple to help you design maps and speed changes. One amazing feature is a procedure that designs it's own maps based on a picture that is entered in .iff format. Think about that, a map based on a politicians face. Frightening!

This amazing utility with manual are available at £29.99 P.P. from:

Shadow Software,
Aaron Fothergill,
Whiddon Valley,
Barnstaple,Devon.
EX32-8NW

If you use Tome, then the control of the characters will be based on Tile Values, this is very easy to implement with the Tile Value facility in both Tome and Tome Junior.

Speaking of which...
as Tome in either version will not run under Amos Professional, there is another way of putting a map on the screen, using the simple program below. It does not however, scroll the map smoothly around the screen.

The information for the screen is put into a string that is concatenated (strung together) into a very long string. Each number represents one of the icons or small pictures that are used in The Black Tarot to make up the picture representing the dungeon.

The numbers following from the rem statement are used to measure out the width. Without the guide line along the top it's easy to get lost.

It will help to remember that the number 2 is a stone wall, that number three is a brick wall and that the space in the string is a floor.

Procedure MAPDO / or any name you like as long as the T$ is global

```
rem     01234567890123456789012345678901234567890123456789012345678901234567890
t$=t$+"2222222222222222222222222222222222222222222222222222222222222222222222"
t$=t$+"2                                                                    2"
t$=t$+"2222222222                                                       2"
t$=t$+"2222222222                                  33333333333333   2"
t$=t$+"2222222222                                  33333333333333   2"
t$=t$+"2222662222                                      33   2"
etc, etc.....
TINY_TAME[T$,50,24]
Return
```

Rather than enter the whole map into the computer by typing and take up a lot of page space, the program Tame and the information can be found on the disk in the main game. It's easy to find using the Search facility found on the upper right of the editor bar in edit mode. You can see at the bottom of the t$, a call to the Procedure

TINY_TAME[T$,50,24]. The procedure passes the name of the string (T$), the width of the map (50) and the height of the map (24). It doesn't matter what size you make your map, as long as the correct parameters are passed over. All the information is then stored in bank 6 so it can be accessed immediately when the game is loaded.

It also helps if you can have a rough sketch of the various icons beside you as you work. If you can dump them to printer from a Paint program in .IFF format, it will speed up the process.

```
Procedure TINY_TAME[T$,W,H]
Rem This program is in the Public Domain!
Rem It was written by Stephen Hill for the Game Maker's Manual (AMOS edition)
Rem and is reprinted with permission of the author
Erase 6 : Reserve As Data 6,W*H+50
Doke Start(6),W : Doke Start(6)+2,H
Rem load map
For M=0 To Len(T$)-1
I1=Peek(Varptr(T$)+M)
If I1=32 Then IC=0
Rem handle numbers (All TILE values  are  ONE LESS than the ico If I1>=Asc("1") and ↵
I1<=Asc("9") Then IC=I1-Asc("1")
Rem handle letters (A=10,B=11..etc)
If I1>=Asc("A") and I1<=Asc("Z") Then IC=I1-Asc("A")+10
If I1>=Asc("a") and I1<=Asc("a") Then IC=I1-Asc("a")+10 Rem check for an error
If IC>Length(2) or IC<0 Then IC=0
Poke Start(6)+M+4,IC
Next M
End Proc
```

This small but efficient procedure can be cut out and used in any of your future programs. As the bank is reserved inside the program, all of the information is saved out with the game. So, once the strings have done their job, they can be deleted. As long as bank 6 is not erased or written to, the information will always be there. If the bank is saved out as "MAP.abk",6 It can be loaded in and processed by TINY_TAME. You don't have to worry about screen size. That has been taken care of during the creation of the bank by the lines...

```
Erase 6 : Reserve As Data 6,W*H+50
Doke Start(6),W : Doke Start(6)+2,H
```


The positions of the walls and floors and the locations of all the objects that you put into the dungeon are saved out as a bank for inclusion into the game 'machine'. These screens will be static, but there is no reason why the adventure cannot be a flip screen game.

Chapter 2

Ideas and inspiration.

- The Celtic Twilight Stories around the campfire and the storyteller in History.
- Suggestions for Game ideas without plagiarising.
- Programming an idea generator on the Amiga.
- Setting large graphics into the game.
- Looking at the example backgrounds from the disk.
- How to wreck the atmosphere of a good game.
- The game of The Black Tarot.

The background to most role-playing games has its history far back in the heart of ancient Britain and Europe. The talented Bronze Age people called Celts brought with them into Britain arts and skills of the highest quality. The Sagas of their gods and goddesses, heroes and villains soon became entangled with the characters and legends of the indigenous natives. As these red-haired settlers expanded into Wales, Ireland and Scotland, battles and great events, weddings and betrayals were remembered and told as stories in which history and fantasy were exaggerated and intertwined. All was remembered and recited on great occasions by bards, those great storyteller-magicians.

Centuries later, when Christianity came to Europe and the keeping of records in writing was developed by the monks, the stories were set down on parchment. In Wales the famous Mabbinogeon is a treasure-house of Celtic history and romance. The Irish have The Yellow Book of Lecan, The First Battle of Moy Tura and The Book of the Dun Cow, written about 1100. In England there is the Romance of Arthur and the Knights of the Round table, recorded and embellished by Goeffry of Monmouth. Even in these ostensibly Christian tales, magic and the pagan Celtic legend appear. In Gawain and the Green Knight, with its tale of the beheading of an indestructible green figure who has a seductive wife, the Celtic lore of Jack O' the Green and the Maiden are not too far below the surface. In the ancient Welsh tales from the Mabbinogeon, the story of the Head of Bran, or Rhiannon and her magic birds hark back to the very beginnings of the spoken word.

In all countries throughout the ages, fables and stories of battles and great events have added to the bank of folklore. Early clay tablet records from Babylon tell of Gilgamesh and his close friend Einkido. Their fabulous journey to find an antidote to a snake bite forms a long story that still rivets the attention, even now. The stories from Arabia and the deserts of the Middle East are as exotic and intriguing as can be found anywhere in the world. Delving into the folk-sagas and myths of the world shows they have a lot to offer the game designer.

It would be almost impossible to read these stories and not come away with an imaginative plot for a unique game or location. There is no copyright on any of the old-folk tales or legends. The stories of other civilisations, Arab, African and Aboriginal also have roots that go back to pre-history. It might be a good move to store plots and ideas in a database on your computer. Any good word processor will do. Strange names and colourful locations can do a lot to give atmosphere to a game.

Sometimes, at some stage in the creation of a game, your brain dries up and ideas stop flowing. This is where an ideas generator comes in useful. On the Library disk is an ideas-generator with a selection of monsters and heroes. The game idea-generator is fairly simple, based as it is on the party game, "Consequences". In Namegen, vowels and consenants are stored then retrieved using a simple formula and then concatenated into a string. The attributes are then generated, adjusted to the character and added to the name$. This is then presented to the player for approval. If accepted, the string is saved out to a folder for use later in the game of the Black Tarot. If the choice is not accepted, Namegen starts all over again. When I started to design the Namegen, there were storage strings and arrays all over the program. This usually means trouble. It was trouble. I pruned most of the arrays and dimension dstrings out and relied on the simple setting of the variables R1,R2, etc. It cured the bugs and worked even better than before.

The design has been kept simple to allow you to hack and change it about. The program source was included rather than compiled, so that you could get into it and make changes. All the data statements hold the relevant names and statistics and can be expanded easily. Remember to change the pointers to the data start label if you have to. If you add or subtract a statement or name there is no need to worry, as the program takes care of the length and number of statements in the data line as long as each line ends with "*". Make a copy of the program and play with the copy for a while. Rem statements are included to help you find your way around.

The code runs independantly of the game and can be used at any time for other programs. It has a few routines that are included in Gamegen. Most of the names it comes up with seem to be suitable for the characters although some are pretty bizzare. The layout of the program is simple enough for you to add your own 'bells and whistles'. Just cut the block out and save to your own disk in ASCII for inclusions into any of your games. If you are feeling adventurous, and you have the extra memory, there is no reason that you cannot

join all the programs together, put them at the start of the game and have the computer generate a completely new game each time you start.

As the team folder holds the players you created, they can take them into the new scenario or leave them 'asleep' and create a new batch of brave adventurers.

The pictures of the players are held in bank 2 as icons.abk and the main screen picture is held in bank 5 as a compressed picture. The memory saving on spacked .IFF pictures is immense as .IFF pictures are compressed to start with. This is an important consideration as graphics can take up a great deal of memory.

A small section from the start of Namegen is listed to show the way the database is accessed and the strings scanned. The full listing with notes is included towards the back of the book.

```
'Dir$="Dh1:BLACK_TAROT_32" : Rem change this to the disc you areworking on
Dir$="Df0:" : Rem change this to the disc you are saving the data on.
```

This may look a little unusual but I use it when designing a game. Each dir$ can be remmed off when needed. This prevents the mixing of created data and programs

```
Screen Open 1,320,256,32,Lowres : Flash Off : Colour 0,0 : Cls 0
Unpack 5 To 0 : Make Icon Mask
```

The screen is opened at the start of the program's creation. If the screen open command is not used at the start and the program is compiled, it will probably crash!

```
Reserve Zone 6
Set Zone 1,7,122 To 27,142
Set Zone 2,37,122 To 57,142
Set Zone 3,70,122 To 90,142
```

Reserving and setting up the zones for the buttons. If a screen zone is reserved before a picture is unpacked, the settings of all the zones are removed. This can cause a lot of grief later on in the program. (Sigh)


```
' If the folder team does not exist, then create one.
If Exist("TEAM")<>True Then Mkdir "TEAM"
'
' start of variable and string dimensioning
NPNTR=1 : NO_OF_TEAM=10
TMNM$=""
Dim SPECNAME$(10),CLASS_NAME$(10)
T=0 : R1=0 : R2=0 : R3=0 : R4=0 : R5=0 : R6=0 : R7=0 : R8=0 : R9=0 : TEAMNUM=0
PIC=1 : PICNUM=4 : Rem the number of picture icons that can be used and the starting
picture number Global TMNM$,NPNTR,V$,C$,N$,SPECNAME$(),CLASS_NAME$()
Global NO_OF_TEAM,T,R1,R2,R3,R4,R5,R6,R7,R8,R9,PICNUM,PIC,TEAMNUM,T
V$="aeiou" : C$="bcdfghjklmnprstvwyz"
Restore SPECIES_DATA : For T=1 To 10 : Read SPECNAME$(T) : Next T
```

You can call your team members anything you like to match a space scenario or a wild west spetacular.

```
' Put names into the dimensioned string arrays
SPECIES_DATA:
Data "Human","Dwarf","Hobbit","Half-Elf","Giant","Elf","Sidhe","Troll","Half-Ork",
"Ork"
```

```
Restore SPECIES_DATA : For T=1 To 10 : Read SPECNAME$(T) : Next T
CLASS_DATA:
Data "Fighter", "Cleric", "Thief", "Madge", "Magician", "Wizard"
Data "Mountebank", "Chieften","Bard","Bandit"
Restore CLASS_DATA : For T=1 To 10 : Read CLASS_NAME$(T) : Next T
N$=""
```

To see "Storygen" in action load Storygen.AMOS from the disk and run. The program will prompt you for a few details and then proceed to compose a story based on what information it has in its memory. A simple electronic bard that will tell stories for you!

GAMEGEN IDEAL GENERATOR

```
Global T,STRY$,L,NAME$()
NPNTR=1 : V$="aeiou" : C$="bcdfghjklmnprstvwyz" : N$="" For T=1 To 10
L=Int(Rnd(3)+4)
' generate a name length, but not too short
```

```
PNTR=Int(Rnd(10)) : If PNTR<4 Then PNTR=False Else PNTR=True 'start to generate
the name of the team member.
While L>Len(N$)
If PNTR Then R=Int(Rnd(4)+1) : N$=N$+Mid$(V$,R,1) Else R=Int(Rnd(19)+1) :N$=
N$+Mid$(C$,R,1) PNTR= Not(PNTR) : Rem flips the number from a positive no. to a
negative no. and vice versa. Wend
NAME$(T)=N$ : N$=""
Next T
```

.....

The dimesion string NAME$(t) now holds 10 names, all lower case.

.....

```
For T=1 To 10
Left$(NAME$(T),1)=Upper$(Left$(NAME$(T),1)) : Rem make the first letter a capital.
Next T
```

.....

use these rem statement lines to experiment and try out various combinations or commands. When you have finished with them, delete them.

```
' For T=1 To 10
' Print NAME$(T)
' Next T
```

.....

STORYTELLER

The above procedure calls on the storyteller section of program to generate a concatenated stry$ from random store of data statements

.....

```
STRY$=STRY$+" One night during a great storm, the house was broken into and an
object of great worth was stolen called The "+NAME$(7)
STRY$=STRY$+". When "+NAME$(2)+" discovered the theft, a great outcry was made
and a reward posted. Hints and whispers of it's location in the depths of "
STRY$=STRY$+NAME$(8)+", the City of Tombs, dark £, began to circulate. Many
people have tried to reclaim the "+NAME$(7)+"but all failed horribly "
STRY$=STRY$+"Now it is your turn! Will you decend into the depths of "+NAME$
(8)+" to bring the "+NAME$(7)+" into the light and claim the great reward "
STRY$=STRY$+"from "+NAME$(2)+"."
```

.....


```
For T=1 To Len(STRY$)
If Mid$(STRY$,T,1)="£" Then STRY$=Left$(STRY$,T-1)+NAME$(5)+Mid$(STRY
$,T+1,Len(STRY$)-T+1)
Next T
```

.....

The above section of program runs through stry$ and replaces the pound sign with name$(5) It isn't really needed but can be useful to rename a portion of a string. The pound sign can be replaced with a selection of symbols that would each allow the replacement of a story.

.....

TIDY[STRY$]

.....

I use Tidy[stry$] a lot to print out neatly in a panel. I think it started life when I was designing adventure text games. I still keep asc listings of these scraps of programs around as they always seem to come in useful, now and then.

.....

```
Stop
'
Rem ..... STRING TIDY .....
'
Procedure TIDY[STRY$]
TX=1 : TY=0 : TXT_WIDE=39 : R=TXT_WIDE
I$=STRY$
R$=I$
Repeat
S$=Mid$(R$,R,1)
If S$=" " Then Locate TX,TY : Inc TY : Print Left$(R$,R-1) : R$=Mid$(R$,R+1,(Len
(R$)-R)) : R=TXT_WIDE Else Dec R
Until Len(R$)<TXT_WIDE
Locate TX,TY : Print R$ : R$="" : Inc TY : Inc TY
End Proc
'
Stop
Procedure STORYTELLER
Restore DAT1 : Gosub DATCOUNT : Restore DAT1 : X=Int(Rnd(L)) : For T=1 To X :
Read T$ : Next T : STRY$=STRY$+T$+" "
```

.....

Datcount is another useful bit of code that you can use when you are designing a game. This removes the need to keep altering the read section of the data input, as long as each section of data ends with a "*". Don't forget to restore the data pointer to the label after using it, as the data will be read off the next line if you do. I keep this as a subroutine as I find using data and subroutines together can cause a certain amount of confusion.

```
Restore DAT2 : Gosub DATCOUNT : Restore DAT2 : X=Int(Rnd(L)) : For T=1 To X :
Read T$ : Next T : STRY$=STRY$+T$+" " Restore DAT3 : Gosub DATCOUNT :
Restore DAT3 : X=Int(Rnd(L)) : For T=1 To X : Read T$ : Next T : STRY$=STRY$+ ↵
T$+" " Restore DAT4 : Gosub DATCOUNT : Restore DAT4 : X=Int(Rnd(L)) : For ↵
T=1 To X : Read T$ : Next T : STRY$=STRY$+T$+" " Goto JUMP DAT1:
Data "Many years ago,","Long ago ","Once, far away,","Centuries ago,","Once upon a ↵
time,","Over the hills and far away","Before my father was born," Data "This tale tells ↵
that once,","Before this race of man walked the earth,","Awhile back,","*"
DAT2:
Data "there lived","I think there lived","there dwelled","I remember there lived"
Data "I have been told, there resided","my Grandfather told me,","*"
DAT3:
Data "in the "+NAME$(1)+" swamp,","in the highlands of "+NAME$(1)+","," by the ↵
mountains off "+NAME$(1)+","
Data "near the "+NAME$(1)+" Sea,","in the "+NAME$(1)+" forest,","in the "+NAME$↵
(1)+" caves,","in the seaport of "+NAME$(1)+","," near the foothills of "+NAME$(1)+ ↵
","," in the far land of "+NAME$(1)+","," in the valleys of "+NAMES(1)+","","*"
DAT4:
Data "a great Wizard called "+NAME$(2)+"."," a Madge of great power named "+NAME$↵
(2)+"."," an evil witch called "+NAME$(2)+"." Data "a Hermit, cold and aloof by the ↵
name of "+NAME$(2)+"."," an Alchemist called "+NAME$(2)+"."","*"
```

```
DATCOUNT:
L=-1 : T$=""
Repeat
Read T$ : L=L+1
Until T$="*"
Return
JUMP:
End Proc
```

In Gamegen the names of the heroes, heroines, monsters, situations and plot ideas can be input by you in data statements and stored in an array. A Name-generator similar to some procedures in Namegen is called and a random selection of each database is pulled out

and knitted into a story. Some of the results will seem a little strange but there will be new combinations that you would not have thought of. The database can be updated by yourself as you come across new sources of information. It is also great fun to play with and discover what comes out of it.

As mentioned in chapter two, the game has been built around the idea of the Tarot Cards. They are reputed to have been around for centuries. Perhaps their roots have many branches, but we know for certain that they surfaced in the Renaissance and were immediately popular with the fashionable in Italy. Alchemy and the early sciences were beginning to flower, and the virtues and vices were more tangible than they are today. At parties and celebrations, people dressed as Fate, Love or Death, would recite stanzas declaiming their nature and the possible outcome of going against their admonitions. The pictures of these mystical personages were painted onto the cards, and there are still some very beautiful examples of these in most museums around the world.

Fortune telling has always been with us and the cards fell into their role of divining tool immediately. As the virtues are reversed, or turned upside-down, they usually reverse the fortune or virtue that is ascribed to them. In the proper pack there are twenty one Higher Arcana cards and the four suits. These are the familiar diamonds, hearts clubs and spades. In the Tarot pack they are the coins or pentacle, cups, wands and swords. In this game there are only some of the Higher Arcana for use within the game. Because each picture on the higher cards is colourful and interesting, they make an immediate graphic impact. Pictures of these are on the Library disk and their virtues and vices are easily read from the database in the program.

The title page for THE BLACK TAROT is already on the library disk for you to use. I used DPaint to create the graphics and then SPACK to pack them onto the disk. Keep in mind that the Loading screen or title page colours can be different from the main game in any combination you like. However, things can go really badly if you mix the palette from one scene to the next. Another thing to keep in mind is the dithering of different colours to give an extra shade. This eats memory, so try to avoid it.

Chapter 3

Role Playing

- How to be a hero (or a right nasty).
- Generating a party.
- The naming of names.
- Worlds and how to make them.
- Thieves, Warlocks and Amazons.
- Down amongst the dead men.
- Hit points, health points and other points.
- Weapons and how to present them.

So you want to be a hero? Well, what is a hero? The Shorter Oxford English Dictionary states that a hero is...

- A name given to men of superhuman strength, courage or ability; favoured by the gods; regarded later as demigods and immortal.
- One who does brave or noble deeds; an illustrious warrior.
- The man who forms the subject of an epic; the chief male personage in a poem, play or story.

This is all very well for mythical heroes, but if they have all the advantages of the above definitions, the poor monsters won't have a chance. No matter how many evil, slimy, horrible, etceteras crawl out of the pit, they will get stomped on before they can say boo! The last definition I think is the closest. In the game there are up to four heroes or heroines, perhaps of mixed race or type with a task to do and an end to reach. These heroes and heroines are to be the subjects of this story, an ever-changing Saga. Even with heroes there are weaknesses. Hercules, the archetypal hero and fighter, killed his children in a fit of madness sent by Zeus. If Hercules wasn't mad before that he certainly was after! Zeus was acting in the role of interfering Games-Master, and being bad tempered with it!

Poor Hercules had The Seven Labours to complete. The tasks set by the games designer can be as tough as those set by Zeus or as simple as getting out of the dungeons alive without losing too many members of the party.

To make the game more interesting, give the characters faults and phobias that are uncovered as the game progresses. In the film Vertigo, directed by Hitchcock, the hero has a fear of heights. He is a vulnerable hero, open to weaknesses as well as strengths. This leads to interesting variations in characterisation, rather than a completely perfect being. The character should be able to overcome or avoid his weakness and improve as he progresses. For example, if the characters in a party must pass over a deep chasm on a narrow bridge, the afflicted member could freeze and cause a certain amount of difficulty, especially if half of the party were still waiting to go. If you want to be a really evil games designer, make the fumes poisonous over a set amount of turns! It is then up to the person playing the game to try out several solutions. As the designer of the game, you know of course, but the player does not and has to try and solve the puzzle. Does another member of the party pick up and carry the unfortunate team-mate or do you fall back on the 'drink courage potion' ploy? Instead of the obvious, keep the player on his/her toes.

Likewise, give your monsters and ghosts weaknesses. True good and evil beings are very rare, but a ghost with a sweet tooth or a goblin falling in love with the leading lady would add a certain amount of humour to the plot. Inanimate objects sometimes have a life of their own. In one game written long ago, (and now mercifully lost) I had the hero followed around the dungeon by an Ivory ball. It was impossible to catch but possible to destroy. Towards the end of the game it led the hero through a tricky maze.

You might not put a trap or nasty in each location without slowing the pace of the game, but try to hint at dangers ahead, a few tastefully arranged bones or skulls can perk up the most boring dungeon. In one of the most recent games, atmosphere is dramatically increased to by adding footsteps and weird noises. The music changes with the action. When a monster appears, the music is aggressive and fast, when wandering around, the music is slow and folksy.

Like the program Namegen, most games start with a character generator where the player can pick a portrait to represent a character and then allot a set amount of points to each virtue or skill. In the board games they are rolled out with dice having various numbers of sides. In computer games they are often randomly generated and then picked out from a menu or a series of lists set in a pleasing graphic screen. Basically, they are still a menu, but disguised and decorated. In some games it is a possible to raise or lower the various points for each character and choose their profession. The original points are randomly generated and given to each character as they are created. I think that this way of creating a character takes it further away from the reality of the game. By the reality I do not mean we leave the world of fantasy, but that magic and fantasy by their very nature need a core of logic and reason. If everything is strange and magical, then it becomes too unreal, unless the person playing the game understands the rules for the fantasy world and abides by them.

The character generator we use is fairly simple, but effective, with the added twist that you won't know what kind of persons you have created until you start to play the game with them. (Or cheat and look at the character file for that person.) The most often used character points are....

[Programmer.. add your own ideas here please with plenty of REMS.]

- Hit points. Ability to take wounds and damage.
- Strength. Ability to kick open doors and fight strength.
- Armour Protects against blows - may be slightly magical.
- Magic. Level of spell-casting and magic.
- Intelligence. Brain power and the ability to memorise.
- Morale. How the character feels about the situation.
- Dexterity. Useful in thieves and fighters.

A basic Class or type of player can be....

- Cleric. Good at miracles wrought by faith. Fair in physical battle.
- Fighter. Good in battle. Moderate to poor at magic.
- Thief. Nimble and good quick fighter. Fair magic.
- Magic User. Good at magic. Very poor with weapons.
- Any that you can think of! Be imaginative and think laterally.

Some games have Alignment determined at the start such as .....

- Lawful. Good and biased towards the causes of right.
- Neutral. Basic average person. Can veer from good to bad.
- Chaotic. Aligned with self interest. Not necessarily evil.

Some types of species are .....

- Human As you would expect.
- Dwarfish. Good fighter. Short but powerful. Poor magic.
- Elven. Good with flight weapons. Good magic.
- Halfling Good fighter. Reasonable all rounder.

Here are some of my interpretations of the above. Feel free to change them to suit you.

Hit points..

The ability to withstand injury and damage. May be tied to the strength pointer and linked with magic and alignment with an advanced class of fighter. The Hit points are used in the battle sequences to determine how the battle is going. The results are then shown in

the slider displays set next to the character. The hit pointer can be increased by magic and potions and may be boosted by treatment from a healer or Cleric, especially after a battle. Conversely, Hit Points are lost in a battle and by drinking a poison potion. Magic traps and dangerous environments, such as poison fogs and swamps may drain the Hit Points. When the Hit Points reach a very low level or zero, then the character will die! Hit points are usually set to a randomly generated, medium level when the game is started. The level is then changed according to the character's race, size and class. A Cleric or Thief have hit points deducted. A Fighter or Dwarf have slightly more hit points than average added to their randomly generated points. Magic users such as Witches and Wizards have a very low Hit Point setting. In the Black Tarot, a ceiling of about 65% maximum is set for these characters, because the balance of the game would be lost if they became nearly invincible later in the game. Hit points should not change too much as the game progresses as the points reflect the type of species of the character. A suit of armour, helm, leather jacket or any protective clothing will obviously help to stop the injuries if worn.

Strength Points..

The energy level of the character. These points are set in the same way as the Hit Points. Dwarfs and Giants have higher Strength Points than other members of the team. Halflings are slightly lower in strength but higher than humans. Fighters are stronger than thieves and Clerics, while again the poor Magic users and Elves are at the bottom. Puny they may be, but they come into their own with brains and magic. Strength points are lost by all classes in battles whether the character is hit or not. Expenditure of energy in fighting, kicking down doors or climbing drains Strength Points. As in real life, sleep or rest can rebuild the Strength points, while the ubiquitous magic potion can restore the flagging fighter. To make things a little more realistic, if a potion is used, give the character a case of the jitters and remove the ability to 'rest'. Potions as any easy option could lead to a cycle of fighting and drug taking, and the leaching away of their specialist abilities. As in real life, drug taking has penalties attached, physically and morally. Magic users casting spells will suffer depletion as well as the fighters. Link the amount of power deleted with their physical size and species type and the power needed to cast the spell.

Magic Points..

These points are generated randomly like all the other points but are more unevenly distributed through the characters and linked to species type and profession. Fighters and giants have very little magical powers, while halflings have a little more. Humans can have average to high, while Elves can get to a very high level. As with physical strength and armour, certain objects can confer magical strength and protection. A magic helmet can confer invisibility on the wearer and a charmed sword increases the ability to strike and parry. However, as we know 'There aint no such thing as a free lunch!'. Energy comes from somewhere and is decreased as the magical action or spell is used. The magic for the implement may come from the earth, the magic user or the sun. In this game the

power for the magic weapons is drawn from the character's Magic Points. If the character is aligned with evil, the weapon could draw its power from the other character's Strength Points. As the game progressed all the other characters could get weaker and eventually die. A clue should be given to let the poor team realise what is happening, else they would all 'perish' without guessing the cause. The Tarot cards give certain magical strengths depending on their picture and who carries it.

Intelligence..

Trolls and most non-player characters have very little intelligence, while human fighters have a little more. In the initial generation of points the fighters are kept well to the back when the brains are given out. Dwarfs and halflings are much better off, and their intelligence levels are about average. Humans can range from the totally dim to bright. Elves are intelligent but their range is limited to 20% added to the average point chosen by the character generator. If the profession of any character is a Thief or Cleric, the level rises slightly. If the character is a magic user the level rises again. A magic Elf has a higher level than a Human Magic User. Intelligence in the game is held at a fairly stable level. It is used in magical battles to determine the ability to cast a spell, and for a pointer to the speed that learning and re-learning spells can take. Intelligence can link with strength in battles and forays. It can also be used to determine the speed at which a thief can pick a lock.

Morale..

Morale is slightly more difficult to use. It can be set instead of generated at slightly higher than 50% and then fluctuate during the game, depending on the success or failure of the individual characters or the team in battle or treasure acquisition. If too low it may determine whether the character deserts and goes home. If high it would boost the energy level in a battle either of arms or magic. The morale flag pointer can be used by the computer in deciding when to throw in another monster or leave a few magic scrolls about to boost the flagging spirits.

Dexterity..

Dexterity levels can be used for all the team and most of the monsters. The ability to wield a weapon nimbly and dodge a blow for fighters is taken from the dexterity pointer for that character. For a thief, the level can never be too high. It takes dexterity to pick a lock or detect a trap set in the vicinity or on a treasure chest. Usually a thief cannot detect a magical trap or curse. If you want to have a magic using thief, do it! Dexterity for magic users doesn't really matter too much as they have (or should have) a higher intelligence level than their team members. Monsters can have a dexterity level as this allows for a more interesting and varied combat routine.

Clerics..

Clerics are a type of religious person, somewhat based on the orders of monks and nuns. You can assign them to any fictitious religious order you care to imagine. As stated previously, clerics are forbidden to use any cutting weapon. They tend to go in for mallets, maces and Morning Stars. They also have a reputation for healing powers.

Fighters.

Fighters are the mainstay of Dungeons and dark bits. They are not terribly bright and eat too much. If too many fighters are chosen, there is a tendency for a lot of hack and slash. That's OK up to a point; but if there is an occasion when magic is needed and there is no wizard, a lock to be picked and there is no thief, you are finished!

Thieves..

Speaking of thieves, in these games the thief is unlike the thieves in our own mundane world. They are more the charlatan or mountebank type of rogue. They are also fairly loyal to the team. If someone's treasure goes missing, it may not be the thieves' fault. Maybe! Thieves are nimble and good at detecting non-magic traps. They are famous for leading from the rear and being conspicuous by their absence in battle. In a tight corner they are good fighters.

Bards..

Based on the Welsh magic all-rounders, they were wise and good in battle. To simpify a complex subject, let us say that their magic system was based on trees and runes. They were (and still are) very good singers and have a phenominal memory. They were the lawgivers of their time and kings and thanes had to abide by their judgement or risk shame and loss of prestige. In role-playing games their music strengthens the team and strikes terror into their opponents. Often their insults were famous and used as weapons to lampoon or give great shame to whomever they chose. They can forecast events through the use of runes and are good to fair in battle. In the game of The Black Tarot, their preferred weapon is a sickle.

Knights..

Usually from a wealthy family, they owed allegience to a King and are found on Errants or Questing for any number of goals. They wear armour which slows them down and reduces dexterity but gain in strength and Hit points. As a horse was the important part of their equipment, the absence of this form of transport in the game is glossed over and the knight walks about with the rest of the team. They only fight if they think the cause is just and obey the rules of Chivalry. This may cause a problem if you want to attack something and the knight says no!

Artisans..

The backbone of any fighting team, able to repair weapons and fix whatever is broken. They are drawn from the general public and may offer to come on the team. They may be of any species and numerous skills. With careful planning they can rise through the ranks to become leader of the team. If Napoleon can do it, they can.

Alchemists ..

These Arcane and introverted folk were the fore-runners of today's chemists and apothacaries. Often found in a jumble of retorts and stills, they can be lured out with promises of gold and the possible discovery of the Elixier of Life or the Philosopher's Stone. As they have to leave most of their equipment at home, they are very good at finding chemicals in odd places and mixing potions and chemical weapons such as Greek Fire and poisonous gasses. They also delight in causing explosions at odd times.

Magic Users

Last but not least is the Magic User. They are the traditional Wizards, Sages and wise men of magic. They have little physical strength but when in battle they can stay out of reach of the enemy in the rear next to the thief and hurl spells. Magic users can be clerics as well, but unless dedicated as a Wizard they can never reach the heights of powerful magic and become a Magi. Most of the team can have a few low level spells like 'Light Fire' or 'Poison Check', but keep these talents at a very low level.

In the game of the Black Tarot, the Alignment is not shown but is an indicator of the character and the way people and events interact with him or her. Call it Karma, and you may get an idea of the way it operates.

Alignment Lawful..

These people are lawful, honest and therefore never thieves. You can't have a lawful thief. They would lose virtue points if they were forced to do something bad or that disagreed with their inner voice of conscience.

Alignment Neutral..

This is the average view of most people in the world. If they can do something good their virtue points go up. If they have to kill, they will not loose many virtual points. The number of virtue points lost depends on their profession, experience and history.

Alignment Evil..

These people crave power and are usually totally unscrupulous. They are often good fighters and their morale goes up whenever they kill anything. If their aims are the same as yours, be it treasure or adventure, they can be of use to the party. They should also be able to understand and defeat more monsters than any other alignment. They are dangerous and do not mind who or why they kill. In the Game of the Black Tarot, this may not always be the best course of action.

In the game of the Black Tarot, the hit, health and other points are shown in a small sliding scale. Hit, Health and Magic, the three important pointers to a character's health are shown to the right of each character. The main weapon is shown under the sliding pointer. All other points, objects in the character's possession, are shown in a large panel that appears on the screen when the small character picture below the main screen is clicked on. Here is displayed a larger picture of the character, and it is here that speech and character interaction takes place. A similar panel is used for non-player characters and monsters that are met in the game. More of that panel in a later chapter.

To the above can be added a great many other skills and points such as .....

Fear of Spiders, heights, fire, claustrophobia
Timid, determined, physically disadvantaged, partially sighted, under a curse, tires easily, is strongly attracted to another member of the party, dislikes or hates another member of the party, is related to another member of the party, or even married to a member of the party.

The character could be generated as a friendly vampire, troll or other unusual travelling companion. The character could be generated as fairly normal to start with but turned by magic into a strange beast or species type at some time in the game. To revert to their original form would add an extra task to the adventure and trigger a range of puzzles to be solved.

The character, if killed in battle, may be dead but not lie down. Not evil but active. I wonder if there could be a ghost character after it was killed? I haven't seen it yet but see no reason why not. Doors would not be a problem for the ghost, hand to hand combat would, but not with the undead, etc.

The opponents can be wide and varied. Role-playing magazines every month describe new and amazing monsters and opponents. A lot of these games have nothing to do with computers and are devoted to the Dungeon Master or board game with small figures. These are a rich source of evil creatures to dot around in your game. Most are of the hack-and-slash type but there are a lot of NPC or Non-Player Characters that interact with your team. In the computer game these NPC can talk to and give advice, help or hinder. All these different angles pull together to add interest and spice to a game and draw the player into the world you have created.

The game we are designing here is based on the traditional game with wizards, thieves, clerics and fighters. The Wizards do not have to be directly named as wizards but could be of either sex . There are witches, High Priests or Priestesses, herbalists, Shamans, hermits, medicine men, bards and all manner of exotic beings from the pages of the world's fables. There are not only clerics but Alchemists, philosophers, druids, defrocked priests, doctors and any order of religious house. Apart from thieves are mountebanks, conjurers, acrobats, actors, people skilled in all the trades that people ply.

The generation of weapons can be as large and as varied as the people that wield them. The weapon usually chosen by magic wielders is a wand or rod, as well as spells. The handling of spells can vary depending on the wishes of the game designer. Some games allow wholesale spell casting, but keep the level of power down and the opponents strong to counter-balance the impact of the many spells. Another way to handle the balance of power is to let the magic user cast a spell and then set aside an amount of time before the same spell can be cast again. Another way to handle the magic casting is to stipulate that the magic user must 'relearn' the spell by sacrificing a certain number of turns if the user wishes to. A self destructing scroll that can only be 'read' once or twice before it bursts into flames is simple and effective.

The other members of the team are easier to kit out. All manner of weapons can be used by any save clerics who are not allowed by their order (and custom) to use sharp edged implements. It's OK to bludgeon an opponent to death as long as there is no bloodshed. Dwarfs and halflings favour axes and short swords, while fighters and larger members of the team can use most of the available armoury. Magic can be used with, or as part of, a weapon, but not to such a high level as pure magic users. There is a trade-off of effectiveness with the smaller members of the team using large weapons and vice-versa. The end result of effective weaponry is to keep the team alive until the end of the game. Jewelled daggers are fine for trading but unless they have some advantage over good plain swords, then sell them.

A small amount of weaponry can be assigned when the team is chosen, in much the same way as the team is randomly generated. The choice will have to be a semi-intelligent one so as not to assign the wrong or inappropriate weapon to each team member. The initial outlay of arms will have to start with a fairly low class of weapon. The team members can then earn, find or buy them as the adventure progresses. The trouble with variety is that you have to generate a picture for each weapon, so that when the attributes and picture of the team member is displayed, the weapons that they have in their possession are displayed alongside each portrait. Likewise, it will be very time and computer memory consuming to generate all the animations for all the characters holding all the weapons, moving in all directions. As long as the weapons are displayed in their panels, or are shown held in the character's hand in the display panel, it will satisfy the requirements of the game

Chapter 4

Here be Dragons! (And other monsters.)

- Monsters and why we need them.
- How to draw monsters on the Amiga.
- Monsters live in Libraries and book shops.
- Sources of inspiration.
- How fiends behave.
- A long list of nasties through the ages.

Monsters have always lurked and prowled around the fringes of man's imagination. They intruded into his nightmares hiding in the shadows, just out of sight. Reality had always been a shifting sliding thing as early man tried to explain illness and madness. Even as children, with the probable security of the family, there were things that frightened us and hid in the corners behind chairs and under the bed.

Monsters are sometimes needed to give shape to the terrible things around us. From death to madness, they embody the intangible and enable us to fight a thing, rather than an abstract concept. As an example; Trolls are large, brutal and fight everything in sight. They are very dim and rely on force to get what they want. They can represent the archetypal school bully or overbearing father. Small Goblins, malicious and devious, could be the smaller brother or child next door who steals and bickers endlessly with his brothers and sisters. The list is endless, and the scope for unloading your anger and fears onto a creature to be destroyed or banished is enormous. You cannot (I hope) kill the bully, but you can fight and triumph over those creatures that most resemble them in the interior war of your game.

To make your monsters real and less like cardboard cut-outs, embody them with features that you remember from your fears. Turn your trolls into school bullies and give them the same qualities that endeared them to the local police. Do your trolls spit and snarl, or hide or jump out? These abstract imaginings can become real and live in the mazes of the dungeon. If they are characters that seem real and nasty to you, they will live and be real in the mind of the people that play your game.

The pictures of monsters can be drawn with the help of any art package and saved out as .IFF screens. I use DPaint on the Amiga and can really recommend it. With the new AGA chip set becoming more prevalent, the adventures are going to be more colourful. With the extra four K Ram in the A1200, the games are going to be BIGGER! Keep the size of the animations approximately the same size for all the team members if you can. Large sprites need a lot of memory if they are to be animated. Trolls, being rocky sorts of creatures could hide, disguised as boulders and then metamorphose into their trollish shape to attack the team. Dungeon walls are usually a drab grey, so give all the monsters a distinctive colour to help them show up.

Once the outline is traced, the sheet of acetate can be taped over the monitor or TV screen and the picture traced with a one pixel wide brush. Colouring in and re-sizing is then very easy. I am assuming that you have worked out the average size of all the characters and monsters. Animating is not so simple. It is made a little easier by having the original sketch in the art package and then cutting and pasting arms and legs. In this book I will not go into the details of animation and its ramifications. That is almost a book in itself. However, it might be useful to keep the number of frames in the animations the same from monster to monster so that the same routine can be used to place them into the game without bothering about different sizes or rates of animations.

There is a bibliography of books at the end of this book which will provide good sources of monsters and other beings. I have listed a few of the obvious ones here to get you started.

The Undead..

Generally a rotten lot. Bad tempered but sociable, they would like you to join their ranks. They are the portion of the monsters that once were human but still retain a physical body. Numbered in their ranks are the zombies, ghouls and mummies. The latter are found in sunnier climes, but they don't seem to like the sun. Mummies are a little out of place in a northern, Celtic type of adventure but can add a dash of the exotic if used with discretion. Frankenstien's monster would seem to fit in with the undead, but is actually alive. He was brought back to life even if his 'Pick and Mix' body kept letting him down.

There are also skeletons, and they are definitely dead. They also have an unhappy knack of re-assembling themselves if they are broken apart in a fight. They must be left to the magic users and destroyed by arcane means. They never leave much treasure behind and always seem to attack in groups. There is a skeleton bob on the Library disk with the label SKELL.abk

Vampires are also undead, but of a much higher class. In fact most vampires wouldn't be seen dead with a zombie! Vampires have a section to themselves later on in this chapter. Ghouls and Zombies are the mainstay of a good adventure game and seem to turn up in most Dungeons and Damp Bits games. Ghouls are mostly European and Eastern while

the Zombie has its roots buried in the Afro-Caribbean culture. Zombies are technically not really undead. Zombies are people that have been secretly poisoned and then buried alive, presumed dead. They are then exhumed secretly and set to work for the poisoner without the benefit of minimum wage protection. They can appear in a game under the direction of an Evil Magician or Sorcerer. If the master of the creatures dies, they wander around listlessly and cause no further damage until eventually they die. Again!

Ghouls are creatures from the graves and vaults of the cemetery. They have unpleasant eating habits and like nothing better than to loiter with intent to eat anything that moves. I do mean anything! They are difficult to destroy, because obviously, they are dead. It is possible to hack them to pieces but their bite is poisonous.

All of the undead can be banished or destroyed by clerics or magic users. Clerics use their faith and the holy symbols of their order to repel the undead and have a different order of spell magic to that of the Wizard, Witch or Madge. Wizards can destroy or drive off the undead but need to be in a slightly higher class than the Clerics. Non magic users in the team can also battle with the undead, but the effort is considerable and drains energy very quickly. When attacked by the undead, let the Fighters in the team move to the fighting front and defend the magic users against damage, while they banish or kill them according to their magic power.

Vampires...

If you haven't heard about vampires by now, where have you been? Hollywood has raised the figure of the bloodsucking, sexy undead to a cult figure. The history of the popular vampire stems from the story by Bram Stoker's 'Dracula'. Vampires shun the sun and can be killed off by a stake through the heart, stuffing them with garlic or tricking them into the sunshine. As there are few sun-beams underground and the first two require the Vampire to stay still for a bit, we shall have to rely on magic, spells of the clerics and the magic users. The cleric's holy symbols can be of any design you choose and these might be sufficient to keep the bloodsuckers away. Spells cast by magic users to kill vampires need to be of a high level. Vampires drain the energy of all members in the party, and this is then added to the strength of the vampire. Vampires can be hurt by any weapon of silver. In the Black Tarot, the card of the Moon increases the power of the fighter and if used by a character with a high magic level, will destroy the creature. The Vampire can also transform into a bat and fly away if the battle looks like going against it.

As this is a game of the imagination, consider a character turned vampire by a bite from an attack of an evil vampire. If the character's Alignment is good, he will try to get a cure. If the Alignment is neutral, then the character could turn against his fellows or could remain a friend and ally. If the Alignment of the character is evil, then there would be a powerful new enemy to contend with and defeat.

A much used source of monster and location idea is the Tolkien Book "The Lord of the Rings." A lot of games have been designed with the aid of this great Saga. Some games are directly based on the sections of the book, others have used the characters from the book and changed the names to try and disguise the source of the inspiration. So great has been the impact of the Book on the Dungeon & Damp Bits scene that it is almost impossible to untangle and remove all traces of it from game design. It is in the sojourn of the fellowship of the ring into the Deeps of Moiria that the story of the Balrog is found. The Balrog was a great, dark monster, wielding a flaming whip. Gandalf in a rear-guard action, grappled and fell into the chasm, to emerge stronger than ever later in the story. You could try that type of disappearance and re-appearance with one of your team. Let there be a battle and disappearance of one of the lead characters in the team. They can then re-appear later in the story when the remaining team are in dire striats, causing great surprise to the embattled team and a triumphant victory over their enemies.

There is on the Library disk a small program that generates monsters rather like the Namegen generates characters and saves them to disk. This means that when the game is run, little memory is lost by storing rows of data statements.

[whoever is coding this, it shouldn't be too difficult if the program is 50% database for type and the rest an amalgam of a namegen and points generator. If I give you the details in pseudocode will this be OK!]

Other monsters can be found hiding in the books at your public library. Greek, Roman and European monsters are both colourful and in some cases, very decorative. There are Harpies, creatures with the tops of beautiful women and the bodies of birds. Hedras with many serpent heads and the famous Medusa, the woman with the snakes instead of hair and the face that turned all who saw it to stone. In the Medusa there seems to be a trace of a Snake worshipping cult that was old when the ancient Greek Civilisation was new. Europe has many creatures, both harmful and helpful to man, among them the Troll and the sometimes friendly Dwarfs from the Scandinavian countries.

These books are usually illustrated with pictures and can help you to get started monster-making. To directly copy a picture that another artist has drawn is asking for trouble if you are going to publish the work. It is a form of piracy called plagiarism and is covered by the Copyright Law. There is no reason why you cannot use the pictures as ideas for your own game so long as they are not direct copies. The Librarian will always help you to find the things you need, and if asked nicely will perhaps offer some advice on where to find the particular creature in the books on the shelves. The children's sections are the richest in monsters and usually better illustrated.

Having had a quick look at the un-dead, we can have a shuddery look at the other things that hide around corners in dungeons and caves. There are the giant spiders, dripping with venom, giant leeches that can leap out of the shadows and snack on your neck. There are

all manner of giant things that while small, look creepy, but when enlarged are horrible. There are giant amoebas, giant insects of all species. Giant rats can be found in the subterranean depths and these are horrible no matter what size they are. They always hunt in packs and inflict a severe bite that needs the quick attention of a healer if infection is to be prevented.

Evil ghosts play their part in this game too. Ghosts have an insubstantial form and a force that paralyses and corrupts when encountered. They are usually tied to one spot and therefore may be avoided. They do seem to guard treasure or some object of value though, so a sojourn into the ghosts locale could yield some object of great worth. As the computer screen seems to be unable as yet to use a semi-transparent pixel, these are best shown in glowing green or a white-green. A pink ghost might confuse the player and lead to confusion generally.

It would be possible to introduce a friendly ghost that warns the players of impending doom, rather like the Irish Banshee. As ghosts are simple minded creatures, the programming can be kept simple. Perhaps the apparition will only appear if there is a cleric or a person of exceptional goodness in the party. It might be possible to have the ghost of a fellow adventurer who had died in battle follow the party around, with mixed results.

Here are a few mythical creatures that are a little more subtle than the usual mob that hang-out on dungeon corners. Most have their origins in European folk-lore and some of their ancestors go back a LONG way.

Sidhe.. Irish fair folk, beautiful, capricious, powerful and souless. They are NEVER to be trusted.

Scarlet Women.. Evil and powerful, Moon magic users. They lure the males of the party to death.

White ladies.. Helpful but do not cross them. Can be cruel.

Will-O'-the Whisp.. Harmless but if followed can lead you to treasure or a trap.

Dames vertes.. Helpful, nature figures. Green and friendly.

Fir Bolg.. Like the Sidhe, but smaller in stature. Shape changers and tending more towards mischievous behaviour than malicious actions.

Sylphs.. Similar to Dame Vertes but of a much younger appearance. Never harmful and can offer help and advice.

Leprachauns.. The well known small people that live all around Ireland. They are small and dark and dress in green. Can be both helpful and harmful. They are full of tricks but do honour their word.

Banshees.. An ancestral ghost that warns of the death of a member the family to which it 'belongs'. It is not harmful.

Bat.. Found in dungeons, they do not attack team members with good or neutral alignment. They do attack the team members with evil alignment and are a great nuisance unless dealt with by a banishing spell.

Vampire bat.. These creatures attack all members of the team and are very dangerous. Apart from sucking blood in a very aggressive way, they can also convert a team member to a vampire. They can be banished and are killed by the usual range of weapons as well as the Card of the Moon. They are also damaged or killed by direct sunlight.

Bandits.. Theye are mostly humans that prey on the team in direct combat. They are not magically dangerous and can carry large amounts of treasure.

Beetle.. This is the LARGE version and it resembles a garden beetle with an attitude problem. They are venomous and look nasty.

Ghost.. Pale green creature with little or no intelligence. They often guard treasure or hover over the spot where they were killed. Sometimes they can speak or warn of coming disasters.

Dragons.. They are the biggest and most dangerous of the creatures that are met with in the dungeon. They also take the most trouble to display on the screen and to make 'real'. They breath fire and eat people. There are many fine examples of dragons to be found, sometimes among the librarians.

Elf.. Similar in form to men, but slighter and more dextrous. They are good with ranged weapons such as bows and arrows but are not physically strong.

Gargoyle.. Artificial creature under the control of a powerful Magic User. If the Controller of the Gargoyles is killed, the creatures revert to static stone statues.

Golum.. Similar to the Gargoyle, this creature is modelled from clay and mud. It is very big. There is only ever one attacking, whereas gargoyles may attack in packs.

Ghoul.. Creatures that live in vaults and graveyards and eat flesh, both living and dead. Their bite is poisonous (it would be wouldn't it), and there are usually several in one location. They do not wander around too far from their home crypt.

Gnome.. A small earth elemental. They are smaller than dwarfs. Dwarfs hate Gnomes. Gnomes try to avoid humans and have knowledge of things metallic, ores and treasures. They are very dextrous.

Goblin.. Mean, nasty and dangerous. They will attack anything that moves and are very nimble.

Green Slime.. Not the speediest thing in the dungeon but a killer in its own quiet way.

Harpy.. A flying creature from Greek history that resembles an undernourished hag. Can flit around the dungeon and cause a lot of trouble.

Humans.. Next to dragons, the most dangerous and the most interesting of opponents to meet. They may range from people wanting to join your group to homicidal maniacs that make Gengis Khan look like a wimp!

Lizards.. These are of the large variety and not to be confused with dragons. They are poisonous. Also in this family are the Fire Salamanders that are neutral in alignment but burn on contact.

Werewolf.. A part time human whose change to wolf is linked to the fullness of the Moon. In real time this is roughly twenty one days. In the game you can set your own cycle. Can join the team as a member or be fought with the Tarot Card of the Sun. If he carries the Tarot card of the Moon, he will revert to wolf immediately.

Medusa.. Another lady of Greek decent, she is so ugly that her gaze changes any man or woman to stone. On other species the effect may be different. To add to the effect, she has a fine head of snakes.

Minotaur.. A Grecian half Bull-man. He is very powerful and can take a lot of damage. I have always felt some sympathy for this creature, kept in the dark at the centre of a maze, because his face didn't fit the courtly circles of Minos.

Ogre.. A large and nasty piece of work. He/she is very aggressive and will attack anything on sight. Creatures are captured and kept in captivity to be used as food. They are damaged by the Sun.

Ork.. Lives in caves and dark smelly places. Strong and aggressive with well made armour. Larger than men and not too intelligent.

Skeletons.. Yes, we all have one, but these are independent left-overs that are dead but won't lay down. Werewolves like them (all those bones!) but they are difficult to kill except by magic. May be under the control of a powerful magic user.

Snakes.. Venomous and deadly. Found in the deepest bits of the dungeon. They come in an attractive range of colours and sizes. They are also not too difficult to draw and turn into sprites.

Spiders.. Not yer actual average bath dwellers but huge, horrible, hairy, venom dripping things! There is usually only one to a level and can probably be found in the centre of a web that the team needs to pass through.

Barrow weight.. Guardians of tomb and barrow treasures, their area is limited to where the treasures and the bodies of the dead lie. They resemble the ghouls or skeletons. They wear the armour of the barrow and are aroused by desecration of the tomb or barrow.

Zombies.. These creatures are under thrall to a powerful magic user. They are still living, yet are mostly dead. They can be killed like normal creatures but are fierce in battle as they fear no pain or death.

Chapter 5

Hack and slash is OK but...

- Interaction with characters.
- When to fight or run.
- The balance of power and the level of difficulty.
- Killing and blood everywhere and how to do it.
- Playing one character off against another.
- Presenting the fights and regulating hit points.
- Wounds and the magical curing properties of Tanith leaves.

In some games, interaction is absent or even minimal. True, you can interact with the character by means of a joystick and direct him (or rarely her) around the screen. The game shows the dungeon and the lead character as if you were glued to a ceiling. This is easy to program and allows the designer to save memory and put all the effort into the game-play. The lead character has nothing to offer being a puppet of the game-player. If there is any form of response to its environment it is crude and direct. Platform type games fall into this category. They have their place in the games repertoire and can be great fun to play, if they are well designed.

Another type of Role Playing Game is mouse driven and more complex. There is one character running around in a labyrinth collecting objects and interacting with his environment in a more sophisticated manner. The view can be looking down on, or viewed from the side. Still popular is a view from the character's eyes and using a 3-D, moving scene of the dungeon. Interaction between Non-Player Characters exists but only in the fashion of 'if you give me an object I will give you directions / an object that you will need later / something boring'. Some of these games are very exiting and absorbing, with music and animations that give you that 'being there feeling'. In the game of The Black Tarot, I have settled for a non-scrolling screen that allows for a type of perspective found in medieval manuscripts. I thought that the style would be in keeping with the subject matter. You cannot scroll a bit-mapped screen that has perspective lines in the design. Well, you can but the result is pretty messy. Also, the non scrolling-games design saves on programming and extra memory.

Multi-character games can start like the first simple games with simple, unsophisticated design and with top down design. But this format has the capacity to be developed into a refined and exiting game genre. Characters can crowd on to the game screen, and have the option to follow a lead character wherever he/she goes. The screen often scrolls to follow the action, and there might be music that echoes the action on the screen. Characters meet Non-Player Characters and information is swapped, pints drunk at local Pubs and the teams have many varied skills and talents. The Games of this genre available now range from the dire to the best ever seen. Objects can be swapped between players and wounds tended. Identification between player and characters can be very close. I know of one game-player who spent many happy hours talking to, tending and directing her players. She was devestated when her creations were killed, even though the character could be resurrected. In the most recent games published, fight sequences are animated and the graphics can be eye boggling! There is a lot of life left yet in this method of presentation.

While on the subject of interaction, fighting (and winning) seems to be the main preoccupation in games. The programmer is always trying to arrive at a perfect balance between the fighter and the opponent and ringing the changes to keep the action bubbling along. If your character dives in at the beginning and attacks every antagonist in sight, or you direct the team into an aggressive stance, there is a strong possibility that you won't last ten minutes. Discretion and the slow building of experience-points will get the team through.

AMOS Basic can never hope to emulate the sophisticated machine coding of the newest Role Playing Games, but it has something that the large corporate company games do not. ORIGINALITY! Originality, and the unique story-line that is the end process of a single mind. Amos can do most of the standard code manipulations, while the more sophisticated and speedier actions can be circumvented with a little thought and low animal cunning.

In the fight sequences the display can run from the most basic through to the graphically explicit. If there are to be lots of pictures and animation, the memory will vanish very quickly. If the game is to be played by yourself on a hard drive and you have one Meg. or more, then you should have enough memory. If you want to sell the game, take into consideration that the buyer may only have a half Meg. machine. These days, most Amiga owners have one Meg. and upwards. Keep the memory requirements down as low as you can, even if there is the extra memory. At the end of the programming you can always top up the action and add a few more lines or pictures.

To display a fight or melee, the simplest way is a statement at the bottom of the screen that says, for example; "You have killed the slimy slug and gain 5 gold pieces!" Not very exiting! The next is a panel that can appear over the action on screen or completely replace it. Depending on who fights what the sequence could appear on the screen in this format:

.....

Gladwyn attacks the Ork.
Gladwyn hits the Ork for 2 damage points.
(Now the Orks turn.)
The Ork attacks Gladwyn.
Ork misses Gladwyn.
(Back to Gladwyn.)

.....

This goes on for a while until the battle is over and someone wins. It can get quite boring if the sequence is repeated for a group and each fights several Orks.

To give the Fight a little more pazazz, try using an idea of the Gamegen to add colour and action to the fight. It will not add that much coding to the game and is fairly low in memory use. Here is another example:

.....

Gladwyn steps forwards and swings his axe at the Ork.
The Gladwyn's Axe hits the Ork and deeply gashes his arm. Blood spatters the ground.
The Ork recovers and advances on Gladwyn. The Ork stabs with his sword. The sword slides off Gladwyn's shield and the Ork steps back.

.....

If we look at the pseudo-coding for the above, the way it works will be made obvious.

Single combat routine
if team member is still alive and hit points and strength points are above a certain level then
weapon$= the name of the weapon the team member is carrying
Team_name$ (the fighter) + (randomly choose a verb$ and concatenate
gender string adds "his"/"her" + weapon$ +" at the "
name of opponent is added
string sent to tidy routine and printed in panel
the result routine
if hit points and strength are bigger than a set level
A verb from the selection of hit names i.e. "cuts", "slashes", "gashes" is chosen and the ↵
members weapon string name is added
if weapon anything other than fists then "Gladwyn's axe slices"
else "Gladwyn fists batters
Added to the string is the opponents name.
An attack number is generated based on his hit points plus strength points plus experience.
An attack number is generated for the opponent based on it's hit points plus strength.

subtract the opponents number from the team members.
Add the hit strength of the weapon. The higher the number the more powerful the weapon.
If the team character's number is roughly equal to the opponent's attack number then ..
Use the results to generate a concatenated string.

If Gladwyn's number is very high then he strikes well and strength points plus hit points are deducted from the Ork. If the Ork's strength points fall to zero then the Ork dies and the result printed.

If the Ork's combined points are running low and are below a set limit then the message is given " The Ork staggers back injured!" or "The Ork dies."

If Gladwyn's hit plus strength points are running low then a number is generated randomly to see if he misses his attack. After the string "Gladwyn's Axe slices " , the words " and misses " is added if the random number is perhaps below a half of the total number set. Use the result of the actions to add or subtract points from both sides.

In this combat routine, the players are ranged on the left side of the screen while the opponant or opponants are ranged down the left hand side of the screen. The team members are selected then their attack style is clicked on. The opponant is then clicked on and the computer generates a reply based on the strength and weaknesses of the opponants. After the team or a particular member has had several goes, the monsters have their bash at the brave intruders.

As the adventurers can Heal, Boost and clerics can call on their Gods to assist them , their strength can go up. The energy must come from somewhere and this can be either their own Gods or their own bodies, to be replenished by resting later on or a special potion, herbal or Alchemical.

I think this looks much better than the prosiac printing out of hit points. If you like hit points then they are all there for you to print out if needed.

[COMBAT]

A hit on the arm might reduce the character's dexterity.
A blow to the legs could reduce their speed.
A strike to the chest could reduce their strength.
A bang on the head would reduce their morale.

In the game of the Black Tarot, the cards can act as boosters for hit or strength points or as weapons in the hands of magic users.

In the pseudo code the weapons were fists, or weapons such as axes and swords. Magic users have roughly the same program section to themselves with spells and castings. The data strings hold different words and phrases but the end result is the same.

If you want to add a form of self-survival to your characters and an unexpected outcome try a few of these ideas.

- Your team member retreats when his combined points fall below a set level.
- Your team member retreats when his combined points fall below a set level and calls out to a team member (the one with the most hit+experience+strength) to come to his/her aid.
- Asks for others to come to his/her aid.
- Calls on his/her Gods for aid.
- Gives a battle cry and gives a suicidal rush!
- Goes totally berserk.

Try to come up with your own ideas and include them in a data line that can be triggered when the battle points of the team member or the opponent fall below a certain point. There is also the option of using an increase in battle points to generate phrases such as "A triumphant grin spreads across (members name$(whoever)) face!"

As you can see, there are many ways to present and enliven a game. You can ignore all of the above or mix-and-match the bits you like.

..... COMBAT .....

The presentation is still in text but has come alive with a fuller description of the fight. The panel can be decorated with whatever designs that you care to add. Try adding pictures of the two protagonists in each corner or an exiting tune to add atmosphere.

If memory allows, the expression can change on the faces reflecting the way their attack/defence is going. Should the enemy be faceless then a static picture will do. You could try putting a smile on a pool of slime! As George Bernard Shaw once said "Try everything once, (except incest and folk-dancing)!"

Another way and probably the best is the graphic display of the fight. However, there is a problem though, in that the memory requirement for the display is enormous. You would need a lot of frames for each character animation, even with the added advantage of bob flipping. The same would be needed for each foe that was met. There is the added complication of the weapon. If one character has swapped his axe for a sword, there has to be another set of sprites of that character holding a sword in memory.

In the game of the Black Tarot, the battle sequences are text with pictures, overlaying the pictures on the screen. The small pictures are shown in the corners and the actions are controlled by buttons at the bottom of the panel. If you have the Amos extension Ctext, the lettering can be any style you like and the lettering would be put on the screen using the special Ctext commands.

Another decision to face is whether their bodies are to remain where they have fallen, or if they are to vanish as soon as the battle finishes. Logic says leave the bodies, game logic says get rid of them as soon as possible to save complicated programming and large arrays to hold the location of various stiffs! A middle course can be to reduce the dead opponent into an anonymous heap of arms and legs and obvious skull. If the opponent was something like a leech, giant amoeba or an acid pool, they can comfortably evaporate. Insects can have a little pile of discarded exoskeleton. This releives the graphic array of holding pictures of different species, wearing different clothes and holding a variety of weapons.

There could be a harmless beasty that roams around and scavenges all the old bits and pieces that are left on the floor. Either that or just remove them when the scene changes. When the characters come back, the piles of bones or whatever will have vanished.

As we have concentrated on death and destruction so much, lets examine the recuperation and resurrection of the team. I wonder if anyone has written a game that deals with the recuperation of the enemy. Somewhere, there must be an Ork's infirmary!

The Cleric, Alchemist and Magic User come into their own here, the Cleric I think being the better oriented towards his fellow travellers. Potions taken from opponents and his range of learned mantras can make good most injuries. I have always thought wizards and Madges fairly self centred, but there are of course always exception to the rules. This is what this book is all about I guess. Bending the rules and looking at the whole game from another vantage point.

Tanith leave have always been my favourite heal-all in adventures. Brought up on Hammer House Of Horror films, I used to watch Mummies and Vampires do their terrible things through gaps in my fingers, deep in the plush of dimly lit Cinemas. I decided at an early age that if Tanith leaves could revive a mummy, they must be fairly powerful! So I have included them in this game along with the Tarot Cards.

A healing potion is usually drunk after the battle is over. As in real life, there is hardly ever a break in the fighting to let you attend to wounds and swig strange liquids from odd and eldrich bottles. If you found a bottle in a cellar, would YOU drink it without examining it? These potions can be tested by any magic user or Alchemist without too great a drain on their magic recourses. In mediaeval times there was a belief that a cup or goblet made of ivy wood would neutralise poison. Never having tried it with poison to see if it worked, I cannot recommend it. In a game though, it could prove invaluable.

There is also the thorny problem of resurrection. In a few games, after the demise of a favourite team member, the body of the unfortunate is lugged to a temple and after a substantial payment, is brought back to life. I never did think too much of that in a game. It seems to be a cop-out. If there must be a way around the problem, have the team member so very badly injured that only a skilled healer at the Temple of St Bart's can heal them. This at least echoes what usually happens in the real world. If the team member is so unfortunate as to 'croak it' in the middle of an exiting game, you can always reload the game from disk, previously saved when you thought the action was going to get sticky.

Chapter 6

Independent characters, drunkards walk.

- Going places, meeting people and having a party.
- Intelligence in non-player characters.
- How to plan independent action in magical beings.
- Lookup charts versus random generators.
- Characters that move under, over and through the scenery.

In the game the Black Tarot, we can assume that the Gamegen has given you a couple of ideas to examine and you have settled on one that appeals to you. You need a task or two to fulfil and a goal or object to reach. This may be the killing of an Evil wizard or the reclaiming of the sacred grail Ofra from the dark depths of the sand clogged City of the Dead in deepest Ophir. This grail, stolen from your kingdom, etc, etc, builds a picture of a destination that will help you to visualise the location. If the adventure is placed in a desert, then the borders of the game can be speckled and sandy coloured. The buildings can be based on the dead cities of Ur of the Chaldees or Petra, the rose red City, half as old as time. They still remain, built to inspire future generations of games designers.

The people and monsters that you meet along the way will have their roots based on the local as well. If the scenario is set in Ancient Atlantis, you can design anything that you like for the characters to wear and dream up any manner and type of monster or hazard. If the adventure is set in Germany in the fourteenth century with knights and foot soldiers, then there are both constraints and advantages. The constraints are the limits placed by logic and History. On the other hand, there is a great body of information detailing the way people thought and the clothes they wore. There are pictures of towns and buildings from the period, the money in circulation, superstitions, diseases and the Political structure of that period in History.

In the game The Black Tarot, the period is very faintly European and of the fourteenth to fifteenth century. There is a knight and a Wizard, in fact the scene is set in a distinctly Dungeons and Damp Bits scenario. When the characters wander about and explore the setting, they will expect to meet and interact with their environment. Character interaction both between the members of the team and monsters is by the simple pop-up panel method. If a character or monster appears and you want a particular member of the team

to talk or interact with it in any way, click on the small picture of your adventurer at the bottom of the screen and the pop up panel will appear. Click on the action and then on the names of whoever you want to interact with. This initialises the course of action needed. Any other actions not included in the program listing can be inserted by yourself in later games.

.....

A few actions are to be...

Fight
Talk with simple click on phrases such as question (elicits hints, tips, history or suitably rude response).
An 'if you give me an object I will give information/another object'.
Included on panel will be large picture of main character and if possible a small picture of the person or monster addressed.]

Rather than referring to people and monsters that appear to the adventurers during the game, I will refer to them as NPC's or Non-Player Characters.

The NPC's met in the adventure need to have a little intelligence, or at least a decent series of responses to questions that are put to them, or situations that arise during the game. It might be interesting to have a very cowardly monster that runs away when spotted. The challenge would be to catch the elusive thing and question it, perhaps by cornering it, or luring it out by a spell or putting out food. A small Gamegen style section can add a bit of interest to the answers received from NPC's and a small look-up data table can hold a list of requests from the NPC's to be carried out, rather in the nature of a small quest. Perhaps a monk or hermit wants a gang of Orks removed from the next dungeon chamber or a prisoner released from a cell. Decisions, decisions!

In a text only adventure game (can you remember that far back?) the description of a NPC met could fill several lines. For example a description could read as; "An Ork blocks your way through the great stone door. The creatures skin is warty and green and it holds a great curved sword that swings back and forth."

Well, that's what you can do with words in a text adventure. But, what about a graphic adventure? The Ork in this game can certainly fight, but the warts are going to be a problem. With the size of pixel being approximately four inches when scaled to the creature, the animated graphic will have to do without them. Our Ork does walk around and can be approached and retreated from. He also 'looks' about at his (or it's) surroundings and 'sees' the other players. The same routine for this action can be used by all the characters and NPC's.

Some NPC's can have more sensitivity than others to the distance that the players are from them. This limitation is set in the listing/module (no. of listing). This Ranges from immediate recognition from a Ghoul to none at all from a slime or acid pool. This range of numbers is stored in a look-up table. All look-up tables take a certain number of bytes. In some games, all the names and hit points are stored in lines of data. It does bring an element of certainty to the game and makes programming easier. The problem is that once the game is played, that's it. You could play the game again trying out various routes, but the responses will become 'samey' and the game start to bore.

It would be possible to have a game using only all random generated game play. This would certainly give variety but the structure of the game would appear flabby and without form. Compromise and common sense is the best. Most randomly generated responses need a sort of look-up table anyway so it's horses for courses. In the end it all comes down to memory use and what you want your program to do. As it is a graphic adventure there is always the need to conserve memory.

If you were wondering about the phrase 'drunkard's walk', used at the beginning of this chapter or a random buffeting of the plot. It also means for me the addition of eccentricities to the behaviour of all the members of the team. I know I have mentioned this before but; try to avoid everything going in a straight line. Keep changing the shape of the dungeon and the colours of walls, at least change them from level to level. If there is to be a confrontation, throw in the unexpected. Do the foes know each other from way back. Have the opponent circle round the team or have several appear from several doors at once. Throw in a tipsy Ork. If the person who is to play the game knows pretty well what is round the next corner or in the next tomb, you have lost them.

When a character approaches a door and begins to go through what does the player see? Does the character suddenly vanish, or does the figure walk under the lintel or arch. As with all computer games, memory restrains intrude into the artistic flow of the game, or putting it another way; the programmer starts to make moaning noises about now. As the game of The Black Tarot is created using Tome Junior for scrolling, it is possible to float a sprite over the lintel area. If the sprite is of a higher number than the character, then the character will appear to 'go under' the exit. Once hidden below the sprite, the character sprite can be removed.

Chapter 7

Artificial intelligence, Graphics and Animation.

- What is possible on an Amiga.
- The loading of data banks as memories.
- Making a character's name and history generator.
- Examples, both simple and complicated.
- Bobs and graphics viewing programs.

A few good pictures can seriously deplete most available memory. AMOS basic has an answer in the several graphic compression routines. The commands 'Spack' and Pack are very effective for squashing a picture in memory, thus saving much needed disk space and loading time. The instructions in the AMIGA manual, (page 276) are very easy to follow. The unpacking is easy too, utilising the AMOS command UNPACK which unpacks both compressed blocks of data.

I have used the Spack routine on the picture for the Namegen screen as the colour information is held with the data. The Pack routine saves a lot more memory as only the minimal amount of picture information is saved. As the picture for the Namegen was the only screen pic needed, it seemed the most logical choice. As the memory size of the picture is variable, small vignettes can be quickly loaded and make a great difference to the overall appearance of our graphic adventure. As the pictures are saved out in .ABK format there is no need to set up and reserve memory as this is done for you.

Below is the complete listing from the character generator with procedures that use various methods of storing, packing and saving to disk.

```
'Dir$="Dh1:BLACK_TAROT_32" : Rem change this to the disc you are working on
Dir$="Df0:" : Rem change this to the disc you are saving  on
Screen Open 1,320,256,32,Lowres : Flash Off : Colour 0,0 : Cls 0 : Unpack 5 To 0 Paste ↵
Icon 12,13,1 : Wait 5
Make Icon Mask
```

The two lines at the top are ONLY used when you are working on the code. If you have a hard drive then this speeds up the design and experimentation. If you have to load in

blocks of data every time you want to debug a routine it can get very boring if the access is floppy only. As the finished game will probably be run from a floppy, there are times when you need to load in data from the disk. All that is needed is to swap the ' character around so that the correct Dir$ is set.

```
'
Reserve Zone 4
Set Zone 1,7,122 To 27,142
Set Zone 2,37,122 To 57,142
Set Zone 3,70,122 To 90,142
```

The button zones are set.

```
'
' If the folder team does not exist, then create one.
If Exist("TEAM")<>True Then Mkdir"TEAM"
```

This procedure is called to check if there is a folder on the disk by the name "TEAM". If there is a folder no action is taken. Else it creates a new one named "TEAM". The Exist command is a useful command at the start of most programs that rely on data held in several folders. The Exist routine could tell you that a certain folder was not present and prompt for permission to create one. That section or procedure can then be deleted from the final program.

```
' start of variable and string dimensioning
NPNTR=1 : NO_OF_TEAM=10
TMNM$=""
Dim SPECNAME$(10),CLASS_NAME$(10)
T=0 : R1=0 : R2=0 : R3=0 : R4=0 : R5=0 : R6=0 : R7=0 : R8=0 : R9=0 : R10=0 :
TEAMNUM=0
```

Well, yes I know that the R1 to R10 variables appear clumsy and take up more room than a dimensioned array, but as the program has a lot of lines that measure and change each stored value, it was easier to have separate variables and to poke them into a bank when each particular character generation had finished.


```
PIC=1 : PICNUM=4 : Rem the number of picture icons that can be used and the starting
picture number
Global TMNM$,NPNTR,V$,C$,N$,SPECNAME$(),CLASS_NAME$()
Global NO_OF_TEAM, T, R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, PICNUM,
Global PIC, TEAMNUM,T
V$="aeiou" : C$="bcdfghjklmnprstvwyz"
' Put names into the dimensioned string arrays
SPECIES_DATA:
Data"Human","Dwarf","Hobbit","Half-Elf","Giant","Elf","Sidhe","Troll"
Data"Half-Ork","Ork"
Restore SPECIES_DATA
For T=1 To 10 : Read SPECNAME$(T) : Next T
CLASS_DATA:
Data"Fighter","Cleric","Thief","Madge","Magician","Wizard","Mountebank","Chieften",
"Bard","Bandit"
Restore CLASS_DATA
For T=1 To 10 : Read CLASS_NAME$(T) : Next T
N$=""
'..... START OF MAIN PROGRAM .....
' The whole program starts and stops with this procedure. It has been designed to be
flexible and can be cut out and re-used with other programs
NO_OF_TEAM=0
Repeat
'..... START OF PROCEDURE LOOP .....
TEAM_NAME_GEN
Until NO_OF_TEAM=4
Stop
'
'..... ...TEAM_NAME_GENERATOR .....
Procedure TEAM_NAME_GEN
L=Int(Rnd(3)+4) : Rem length of names
PNTR=Int(Rnd(10)) : If PNTR<4 Then PNTR=False Else PNTR=True
'start to generate the name of the team member.
While L>Len(N$)

If PNTR Then R=Int(Rnd(4)+1) : N$=N$+Mid$(V$,R,1) Else R=Int(Rnd(19)+1) : N$=
N$+Mid$(C$,R,1) PNTR= Not(PNTR) : Rem flips the number from a positive no. to a
negative no. and vice versa.
Wend
TMNM$=N$ : N$=""
: Rem now team member's name is n$
```

The program lines above select a constant and add them to n$ until the length of the string is equal to L.
The n$ is then put into TMNM$. n$ is then cleared ready for the next name to be generated.

```
.....
Left$(TMNM$,1)=Upper$(Left$(TMNM$,1)) : Rem make the first letter a capital. '
'..... TEAM_POINTS_GENERATOR.....
'
Rem '"Human", "Dwarf", "Hobbit", "Half-Elf", "Giant", "Elf","Sidhe", "Troll","Half- ↵
Ork", "Ork"
Rem '"Fighter","Cleric","Thief","Madge","Magician","Wizard""Mountebank", ↵
"Chieften","Bard","Bandit"
.....
```

I just put the above in the program to remind me of the choices, again, in the final program these can be deleted to save space.

```
.....
'
NOWAY:
Rem points are generated for human then altered according to species R1=Int(Rnd(9)+1)
R2=Int(Rnd(9)+1)
R3=Int(Rnd(9)+10)
R4=Int(Rnd(9)+10)
R5=Int(Rnd(9)+10)
R6=Int(Rnd(9)+10)
R7=Int(Rnd(9)+10)
R8=Int(Rnd(2)+1)
R9=Int(Rnd(1))
.....
```

The above are starting attribute numbers for the characters that are to be presented on the choice screen. Most of them will certainly be changed on the following program lines

```
.....
Rem .. adjust for species etc.....
Rem .....DWARF.....
If R1=2 Then R3=R3+5 : R4=R4-3 : R5=R5+7 : R6=R6+2 : R7=R7-10
.....
Rem R1 .....The character is of the Dwarfish Race.
Rem R2 .....They are quite strong so boost strength.
Rem R4 .....But not too bright.
Rem R5 .....They can take a lot of damage.
Rem R6.....They are both nimble and dextrous.
Rem R7 .....They are not very good at magic.
Please tinker with the settings and adjust to suit your ideas.
```



```
Rem .....HOBBIT.....
If R1=3 Then R3=R3-2 : R4=R4-5 : R5=R5+5 : R6=R6+5 : R7=R7+3
Rem .....HALF-ELF .....
If R1=4 Then R3=R3+1 : R4=R4+5 : R5=R5-5 : R6=R6+8 : R7=R7+6
Rem .....GIANT .....
If R1=5 Then R3=R3+7 : R4=R4-7 : R5=R5+8 : R6=R6-7 : R7=R7-10
Rem .....ELF.....
If R1=6 Then R3=R3-7 : R4=R4+7 : R5=R5-9 : R6=R6+9 : R7=R7+8
Rem .. .....TROLL.....
If R1=8 Then R7=0
Rem .. ALIGNMENT.....
If R1>6 Then R8=3 : Rem from sidhe up they are all soul-less
If R1<7 Then R8=Rnd(2)-1
Rem ... sort out who can do what
If R1>7 and R2=9 Then Goto NOWAY : Rem noway can that lot be a bard
If(R2>2 and R2<8) and R1>7 Then Goto NOWAY : Rem they cannot be magic
rem users If R8<1 Then R8=2
' Now put all the altered attributes in the right dimensioned string
'
'..... Just to remind you of the attribute numbers.....
'..... delete when the program is finished .....
'SPECIES=R1
'CLASS=R2
'STRENGTH=R3
'IQ=R4
'HIT=R5
'DEXTERITY=R6
'MAGIC=R7
'ALIGNMENT=R8
'GENDER=R9
'PIC=NO. OF PICTURE CHOSEN
'
Paper 1 : X=15
Ink 1 : Bar 118,20 To 300,175 : Rem used to clear the attribute screen
Pen 21 : Print At(X,3);"Name    "+TMNM$
Pen 18 : Print At(X,4);"Species  "+SPECNAME$(R1)
Pen 18 : Print At(X,5);"Gender   ";
If R9=0 Then Print"Female" Else Print"Male"
Pen 29 : Print At(X,6);"Class    "+CLASS_NAME$(R2)
Pen 20 : Print At(X,7);"Alignment ";
If R8=1 Then Print"Good"
If R8=2 Then Print"Neutral"
If R8=3 Then Print"Evil"
```

```
Pen 29 : Print At(X,8);"....."
Pen 24 : Print At(X,9);"Stength   ";R3
Pen 27 : Print At(X,10);"Hit PointS ";R5 Pen 20 : Print At(X,11);"IQ        ";R4 Pen 18 :
Print At(X,12);"Dexterity  ";R6 Pen 25 : Print At(X,13);"Magic     ";R7 Ink 30
'.....Start of the choice loop .....
ZLOOP:
Wait 5 : rem if there is no wait, the loop whizzes around several times.
If Mouse Key=0 Then Go to ZLOOP : Rem keep looping until a zone is entered and a ↵
mouse key pressed.
 If Mouse Zone=1 Then CHOOSEPIC
If Mouse Zone=2 and R10=0 Then Boom : Goto ZLOOP
.....
```

The Boom sound effect is called to prevent a character being saved without a picture number in variable R10.

```
.....
If Mouse Zone=2 Then TSAVE : Goto FROG
.....
```

I always use frogs for jumps!

```
.....
If Mouse Zone=3 Then Goto FROG
Goto ZLOOP
FROG:
End Proc
'.....
Procedure TSAVE
Pen 31
Locate 15,15 : Print"You have chosen"
Locate 15,16 : Print TMNM$+" the"+CLASS_NAME$(R2) : Wait 40
'
Erase 15 : Reserve As Data 15,40 : L=Start(15)
.....
```

Bank number 15 is cleared and set to 40. Even if there are not forty numbers to save, you may wish to add other attributes in future and the memory overheads on this method are very small. I originally tried random access file saving in a long string, but this proved too fiddly. Len Tucker from Totally Amos suggested an alternative, and this I have used as the method is very efficient and easy to program. As all the data is poked into the bank, there is no problem loading in as the file is in .ABK format. This means that upon loading the saved data from disk, the right amount of memory is set aside and the start of the data is set automatically by AMOS. You must reserve the right amount of data to start with though, and you must find out the start of the bank using" L=Start(The number of your bank)", so that the data can be poked in.


```
.....
Poke L+10,R1
Poke L+11,R2
Poke L+12,R3
Poke L+13,R4
Poke L+14,R5
Poke L+15,R6
Poke L+16,R7
Poke L+17,R8
Poke L+18,R9
Poke L+19,R10
Save"ATTRIBUTES/"+TMNM$+".ABK",15
.....
```

I have used the name of the character saved as the filename and concatenated the folder name with the team name string (TMNM$) and then added the .ABK and bank number on the end. As some other pictures and icons are saved in .ABK format, it seemed prudent to keep them separated from the others in their own folder.

```
.....
Inc NO_OF_TEAM
End Proc
'..... CHOOSEPIC .....
Procedure CHOOSEPIC
Inc PIC
.....
```

If PIC>PICNUM Then PIC=1 : rem keep cycling through the selection of pictures. There is the problem of gender choice. It is possible for the gender flag R9 to signal a sex and have male/female pictures presented as needed. All that would be needed then would be the segregation of the sexes into separate folders.

```
.....
R10=PIC
Paste Icon 12,13,(R10+1) : Wait 5
End Proc
.....
```

The variable R10 is then set to the picture number and the picture icon pasted over the frame.

The background-inserts and all the graphics used in titles and vignettes are stored as .ABK files. Use PICVIEW.AMOS to have a look at them and save them out as .IFF files to change and play with. There is a screen with Gothic letters, a background to experiment with, and a folder of faces and sprites. Some of the smaller pictures are of

graphic buttons used in the game. There are also some animation screens for you to load into the AMOS sprite editor or Aaron Fothergill's Spritex and perhaps include in your own game. As stated before, there are no hard-and-fast rules about what goes where. There is enough material in this book and disk for several games.

Chapter 8

Not just a load of old modules.

- Laying out the program so that it makes sense.
- Going sub and passing variables.
- Grouping structures.
- Labels and where to stick them.
- Beginning the game with an introduction.
- The loading screen and intro as a separate program.
- Chaining.

Amos has a very useful way of making the editing and de-bugging easy called folding and unfolding. When the procedure has been set up and the coding finished, the program can be folded, leaving only the Procedure NAME_OF_PROCEDURE showing. As AMOS Basic starts at the beginning of the program and scans down from the start to find the procedure, it makes sense to put the most often used procs. at the beginning and the least used at the end. It won't always work out that way, but try to do it if you can. The other thing to remember is that if you try to grab as a block a folded procedure to save out as ascii, all you will get is the procedure header.

The AMOS manual explains the details of passing variables using the square brackets so I won't go over that again. I myself don't use that method often, preferring to turn every variable and array into a global. There are a few square brackets passing local variables here and there though. Using the "On number Proc Proc1, Proc2" method of accessing procedures disallows the use of variables in square brackets. Choose whatever method suits you. After a while you will settle on your own style.

In basics such as STOS on the ST, the lines have to be numbered and as there are no procedures, all jumps are to line numbers using GOTO, RETURN and GOSUB. In AMOS structured basic the procedure makes the program easier to read, even if there is a very small loss of speed. The other advantage with modules is that they can be cut out, (remember to unfold them) and stored in .ASC or .AMOS to be used in any other program you design.

Sometimes there needs to be several routines in one procedure as shown in the hypertext program listed later in the next chapter.

This is where labels come in useful as there are no line numbers.

It is also possible to have one program calling another so that the whole game can be run in several "chunks". The listing below is a case in point. The introduction is very heavy on graphics memory and special effects using AMAL but is used only once and then never used until a new game is run. This method is ideal to present a really impressive start without jeopardising the memory used in the actual game. As the last screen picture is kept on screen until a new command from the newly installed program changes it, the transition from one program to another can be made very smoothly. As there is to be little delay as possible in loading the intro, the packed screen is kept in bank 5 and unpacked at the beginning of the program.

The Run command deletes the resident program as the new program is loaded. This will remove all variables and pointers and (apart from the screen) it is literally a new program. There is nothing to stop you saving out to disk or ram, a bank that has had the variables poked in and then immediately re-loading them in again. Ram is neater and almost instantaneous.

```
.....
'..... LOADING SCREEN AND INTRO .....'
'..... STAND ALONE PROGRAM .....'
Dir$="df0:" : Rem change this to the disk the program will run from when developing, ↵
then delete when the game is finished.
Screen Open 0,320,256,32,Lowres : Cls 0 : Hide : Curs Off : Flash Off
Load "lscreen.abk",5
.....
rem use this when developing the program then delete on completion. The'X=Free' is a ↵
neat way of cleaning up the memory pointers.
Unpack 5 To 0 : Erase 5 : X=Free
.....
Double Buffer
Load "cards.abk" : Rem load the card animations as bobs
.....
```

As we are using Amal to animate the cards, the commands need to be entered into a string. When I first started with AMOS I avoided the Amal because it was so unlike BASIC and the early compilers crashed on some of the commands. Now I use AMAL when I can because of its Multi-tasking ability.

A small description of the first line is given. The AMAL language is fully described in a clear and concise way in the AMOS Manual starting on page 176. The easiest and most frequent mistake (I find) is the using of lower case letters instead of upper case letters for the instructions.

```
FLIP1$=" L X=254 ; L Y=51 ; M 0,0,20 ;  A 1,(7,6)(6,6)(5,6)(4,6)(3,6)(2,6)(1,6)(1,6) ↵
(2,6)(3,6)(4,6)(5,6)(6,6)(7,6)"
```

.....

FLIP$= ..... The string name, can be any name you wish except for reserved AMOS words. If you want to use a reserved wording a word like DOWN which keeps separating into Do and WN , put an underline in front of the word, _DOWN. That fixes it!

.....

L X=254 ..... L stands for Let and is used to set the X variable to 254 . This moves the bob to X on the screen. The semi-colon ; is used as a spacer such as a : is used in basic programming. The L can be entered as Let as long as the capitol L is used. You could enter Ligwibble. The effect would be the same, all the lower case letters are ignored.

.....

L Y=51 ..... Moves the bob down Y pixels.

.....

M 0,0,20 ..... The M is the move command. It can be read as Move X,Y,steps . In this command I have no X or Y and use the Move command as a sort of Multi wait

.....

A 1,(7,6)(6,6)(etc,) .... Anyone using the original STOS will recognise the old Anim sequence string. Animate bob 1 using sprite picture 7 and waiting 6 times before going to the next command. Well not really. It is almost the same, but now the A stands for animate and the 1 is how many times the animation is played. A 0 gives an endless loop through the string. The figures in brackets have the same function as before.

.....

```
CARD1$="(13,6)(12,6)(11,6)(10,6)(9,6)(8,6)"
CARD2$="(13,6)(18,6)(17,6)(16,6)(15,6)(14,6)"
```

.....

```
DWN$=" M 0,97,100 ; Move 0,0,10 "
```

This DWN$ just moves the tumbling card picture down 97 pixels in 100 time slices and then does nothing for 10 time slices.

.....

```
For T=1 To 3 : Channel T To Bob T : Next T
Bob 1,-100,-100,1
Bob 2,-100,-100,1
```

setting the X and Y co-ordinates to -100 sets the bobs on and well above the screen out of sight.

.....

DEAL$=FLIP1$+CARD1$+DWN$
The two animations of the cards turning are concatenated and put into the DEAL$.

.....

Amal 1,DEAL$: Amal On 1
The Amal command is pointed to bob 1 and the string that holds the instructions that it will follow. The command Amal On 1 then starts the sequence off.
Amal 1,DEAL$: Amal On 1

.....

Wait 180
As there is no game play in this section and no other action is taking place I used a programmed wait to allow for the sequence completion before going on to the next Amal sequence.

.....

DEAL$=FLIP1$+CARD2$+DWN$
Amal 2,DEAL$: Amal On 2
'
Wait 180
'
Erase 1 : Rem Use in the last line while developing to save memory when saving to disk
X=Free : Rem garbage collection

.....

'Run "MYPROGRAM"
The AMOS command Run will look at the disk pointed to by the DIR$ command and load in then run the next AMOS program. I have remmed out this line because if the program is run during development, it vanishes when the next program is loaded in.

Chapter 9

- At last, some of the program procedures.
- Credit where it's due.
- Music and the installing of modules.
- Grouping structures.
- Labels and where to stick them.

Well, here it is at last, most of the modules with some of my routines. To be honest, I will admit that some of the trickier routines were passed on to me by Aaron Fothergill of the Amos Club and Len Tucker, who publishes and distributes Totally Amos. This is a very useful and interesting magazine, and is packed full of routines, advice and news. When you are starting Amos there is so much to find out and learn, and even when you are competent, there are always problems that escape you and solutions available from a fresh viewpoint. The Totally Amos Disk Magazine always has the latest news in upgrades, reviews and products. Information on these two very helpful people and their clubs are on the pages at the back of this book.

At the very beginning of the program is a rather odd line that plays the opening fanfare. It uses a module from Technical Fred in the 'c' library on the boot disk to play a tune. There are not many Sonix Music programs around now, but I still have a soft spot for it as I find entering musical notation easier than entering numbers in a grid. There are many music player modules around in PD Libraries that can be used to generate music fitted in to your game. Amos has a few converters on the extras disk. There are also many in existence in the PD Libraries and Len Tucker has a large selection of them in his PD Library as well.

.....

```
EXECUTE["sonix_play > NIL: * i=df1:instruments df1:scores/fanfare"]
Dir$="dh1:BlACK_TAROT_2"
```

The instruments and scores are stored in files on the boot disk. If you want to play the game from hard drive, the module can be copied over to the hard drive 'c' folder.

.....

When I started writing this program, there was no idea of a book to go with it . I originally used Ctext and the Latest Tome map maker to write it. As the programs are not PD and are only available from Aaron Fothergill's Shadow Software, I had to take out some bits and change others. I like Ctext. Ctext allows you to design any font, in as many colours as you like and store it with the program you are writing in the banks. The original font was a little smaller than the system font used in the game, and I could therefore put more information on the screen. The Tome I am using is Shadow Software's Tome Junior and is a stripped down version of its big brother. Highly recommended! The new improved version is available at a very reasonable price and speeds up game speed and development. It also saves a hell of a lot of memory that can be used for graphics and other things.

The variables are now dimensioned and the globals set.

```
Dim NAME$(4),OPPNAME$(6),_ATTR(4,40)
Dim OBJTEAM(4,22),CLASS_NAME$(10)
Dim SPECIES_NAME$(10)
DimWRD$(10),WRD(10),MESS$(15),OBJNAME$(25)
Dim OPPONENTS(6),OPP_ATTR(6,6)
Dim ENEMY(4,6) LEADER=1 :  OLEADER=0 : SPEAK=False :  TAKE=0 :  PNTR=0
LOOK=False  SWORD=False  : X=0 : Y=0 : FILE$="" : DV=0 : RV=0 : LV=0 : UV=0
:T=0 : BATTLE=1
MX=96 : Rem map start position X for leader
MY=128 : Rem map start position Y for leader  TX=0 : TY=0 : TV=0 : R$="" : Rem text
variables
HX=0 : HY=0 : FX=0 : FY=0 : CNTR=1 : Rem map handle vars
GM=1 : GTIM=7 : OPPONENT=1 : Rem monster timers
M=0 : A=0 : MZ=0 : MP=0
Global M,A,MZ,MP
Global A$,Z,Q,A,WRD$(),WRD(),MESS$(),TMP,TX,TY,R$
Global HX,HY,FX,FY,MX,MY,CNTR,PNTR,T,LOOK,SWORD,BATTLE
Global WALK$,RWALK$,DWALK$,UWALK$,NAME$(),_ATTR(),OPP_ATTR(),  ↵
ENEMY()
Global TV,DV,RV,LV,UV,LEADER,OLEADER,SPEAK,FILE$
Global RGMARCH$,LGMARCH$,X,Y,GM,GTIM,OPPONENT,OBJNAME$()
Global SPECIES_NAME$(),CLASS_NAME$(),TAKE,OBJTEAM(),OPPONENTS()  ↵
, OPPNAME$()
```

I have included all the variables here even though it will look confusing at first glance. Variables are the life-blood of the program, taking information and data between modules and routines. This area of the program is always being consulted and added to as the game progresses.

The opponents names and team names and classes are read in to the various arrays along with other data.

```
OPPDATA:
Data "Skeleton","Wraith","Troll","Ork","Ghoul","Mummy"
Restore OPPDATA : For T=1 To 6 : Read OPPNAME$(T) : Next T
SPECIES_DATA:
Data "Human","Dwarf","Hobbit","Half-Elf","Giant","Elf","Sidhe","Troll","Half-Ork",  ↵
"Ork"
Restore SPECIES_DATA : For T=1 To 10 : Read SPECIES_NAME$(T) : Next T
CLASS_DATA:
Data "Fighter","Cleric","Thief","Madge","Magician","Wizard","Mountebank" "Chieften"  ↵
,"Bard","Bandit"
Restore CLASS_DATA : For T=1 To 10 : Read CLASS_NAME$(T) : Next T
```

The team and enemy bobs are put on the screen well out of sight, and will wait here until called down by the Amal commands.

```
Bob 1,-100,-100,6
Bob 2,-200,-100,6
For T=1 To 15 : Channel T To Bob T : Next T
```

The channels are apportioned out for the Amal commands.

The set-up procedure is called. From now on the program gets very busy. Rather than include the whole program and all the modules, I will cover the SETUP and the games loop, while the many modules will have a small description or be omitted.. The disk that comes with the book has the complete listing and the program can be dumped out to printer using the 'Get Block' and save as ascii command, available from the menu bar.

```
SETUP

Bob Update Off
Synchro Off : Map Handle Init : Screen 0 : Double Buffer
'

ZSET  : Rem the last and best place to put a set zone command
MAKE_LEADER
MAKE_OPPONENT
Bob 1,32*5+6,32*3-4,6 : Rem first char start pos
Bob 2,-100,-100,
Screen Show 0 : Map Update On
'•••••••••••••••••••••••••••••••• GAME LOOP ••••••••••••••••••••••••••••••••
Repeat
```

```
MAPMOUSE
'.....
```

MAPMOUSE puts the little white square around the point that the mouse pointer is clicked on, This routine is disabled when the pointer is off the map. The place on the map where the mouse pointer is clicked on is turned into map co-ordinates.

```
'.....
MZ=Mouse Zone
```

if the drop command is clicked on, it takes you to the LOOK procedure as the square brackets are not allowed in an On MZ Proc call.

```
If Mouse Key=1 and MZ<>0
On MZ Proc TEAM1,TEAM2,TEAM3,TEAM4,SPEAK,TAKE,BRIBE,LOOK,  ↵
SWORD,GREET,LOOK,SPELL,MAIN_SCREEN
.....
```

If you run into a crowd of opponents, this routine takes you to the battle procedure which sets up the opponents and then on to the fight module 'sword'. Crude but effective. The problem here being that there is a zone already covering the whole map. One idea might be to check when the mouse is > than the X start of map and < X end of map, and then do the same for the Y co-ordinates. As the loop would have to keep reading the information on every loop, it will slow down the action a bit. If this system was used then the zone used for the map could then be set around the central character. Decisions, decisions.

```
If Bob Col(1,2 To 10)
GO_BATTLE
SWORD
End If
.....
```

The four lines below are used by Tome Junior to put the map on the screen, and to scroll the tiles smoothly. These variables are also used by the various procedures to find out where you are on the map and to place the opponents. No easy thing to do as the map is constantly scrolling and they must follow a pre-determined path, even as you 'move' around the dungeon.

```
HX=Map Hx()
HY=Map Hy(MY)
FX=Map Fx(MX)
FY=Map Fy(MY)
'
```


The directions RV, LV, UV, and DV were processed in sequence, one after the other in MAPMOUSE. This tells the leader how far he/she can walk in two directions. RIGHT_MOVE, LEFT_MOVE and the others take care of permitted movement, walls in the way and tile values such as traps and triggers for events.

```
'.....
If RV>0 Then RIGHT_MOVE : Goto FROG
If LV>0 Then LEFT_MOVE : Goto FROG
If UV>0 Then UP_MOVE : Goto FROG
If DV>0 Then _DOWN_MOVE
FROG:
'..... The start of the Map Processing Section .....
'
Screen 1
Map Handle 1,HX,HY
Screen 0
Limit Bob 80,0 To 320,172
Synchro
Screen Copy 1,FX,FY,240+FX,172+FY To Logic(0),80,0 Bob Draw
Screen Swap
Wait Vbl
Until False
Stop
'.....
Procedure SETUP
.....
Clear out all the banks

Erase 1 : Erase 2 : Erase 6 : Erase 8
.....

Load "GRAPHICS/square.abk" : Rem the mouse pointer square
Screen Open 1,320,200,32,Lowres : Get Sprite Palette : Colour 0,0 : Flash Off : Cls 0
Curs Off Screen Open 0,320,200,32,Lowres : Get Sprite Palette : Colour 0,0
Flash Off : Cls 0 : Curs Off Screen 0

Load "GRAPHICS/PANELS.ABK" : Rem panel icons for faces, buttons and fight panel.
.....
```

If Make Icon mask is not called there is a chance that the icons will have an opaque border around them. If all the icons are on a 16 pixel boundary then there is no need for Make icon Mask, and a little memory will be saved.

```
Make Icon Mask
'
' ..... PASTE IN PANELS .....
Hide all the activity until you are ready.
Screen Hide 0 : Screen Hide 1
```

.....

Rather than have one big panel with four repeated sections, the same panel is pasted four times and then later grabbed as a block.

```
For T=0 To 200 Step 50
Paste Icon 0,T,2
Next T
Paste Icon 78,172,1 : Wait Vbl : Rem button selecter '
'..... TEAM PICTURES IN PANEL.....
TLOAD : Rem load team pics and attributes .
'
'..... THE SMALL PANELS .....
Paste Icon 200,50,1 : Get Block 11,200,50,48,46,1
Paste Icon 0,2,3 : Get Block 3,0,0,40,50
Paste Icon 0,52,5 : Get Block 4,0,50,40,50
Paste Icon 0,102,7 : Get Block 5,0,100,40,50
Paste Icon 0,152,9 : Get Block 6,0,150,40,50

'.....IF THEY ARE DEAD WHEN YOU LOAD IN .....
'
If _ATTR(1,0)=0 Then Put Block 11,0,2
If _ATTR(2,0)=0 Then Put Block 11,0,52
If _ATTR(3,0)=0 Then Put Block 11,0,102 '_ATTR(4,0)=0
If _ATTR(4,0)=0 Then Put Block 11,0,152
'
'..... TEAM PICTURE PASTE, GET AND CLEAR ....
```

This just pastes the pictures pointed to by the _attr(leader,picture) and grabs them into blocks. The icons are erased later to free up the memory.

```
Paste Icon 100,50,2 : Get Block 7,100,50,80,95
Paste Icon 100,50,4 : Get Block 8,100,50,80,95
Paste Icon 100,50,6 : Get Block 9,100,50,80,95
Paste Icon 100,50,8 : Get Block 10,100,50,80,95
```



```
' load in the opponents mug shots in small frame
Erase 2
Load "opponents/icon2.abk",2 : Load "opponents/icon1.abk",2

Load "opponents/fight_panel.abk",2
Bank Swap 2,10
' the opponents pictures are now stored in bank 10
' as the bank two space is to be used for the tome map icons
Get Block 1,0,0,80,200 : Rem get the character panel
Get Block 2,81,172,239,28 : Rem get the button selector
Cls 0 : Put Block 1 : Put Block 2
```

.....

SET_ATTR draws the colour bars next to the team members pictures. The routine is quite useful and can be re-used in many games and programs.

```
SET_ATTR
```

.....

The Tome Map and the tiles (icons) that go to make up the map are loaded.

```
Load "GRAPHICS/TEMPMAP_2.abk"
Load "GRAPHICS/TEMPTILE_2.abk" : Rem load icons for tiles into bank 2 over the
panel and team picture tiles
Load "GRAPHICS/TEMPVALUE_2.abk"
'
```

.....

Tell AMOS the size of the tiles and how big the viewing area is and where it will be. In this game all the work is done on a hidden screen one then the map is copied with a screen copy 1 to 0.

```
Tile Size 16,16
Screen 1
Map View 0,0 To 240+16,200
Map Do 0,0
Screen 0 : Wait Vbl
Bob 15,-100,-120,1 : Rem screen square indicator stored off-screen
'
'
Reserve Zone 15
For Z=1 To 15
Read A,B,C,D
Set Zone Z,A,B To C,D
Next Z
Data 1,4,75,47
```

```
Data 0,52,74,96
Data 0,102,75,147
Data 0,152,74,197
Data 91,176,114,196
Data 120,175,142,195
Data 148,176,169,196
Data 174,176,197,196
Data 202,176,224,196
Data 229,175,252,196
Data 257,176,279,196
Data 284,176,307,196
Data 78,4,318,169
Data 80,4,84,8
Data 82,5,88,9
```

'The name or description of all the objects available.

```
Restore OBJNAME
For T=1 To 20
Read OBJNAME$(T)
Next T
OBJNAME:
Data "A clear mind potion"
Data "A steel sword."
Data "Boost energy potion."
Data "Give extra strength."
Data "Boost magic powers."
Data "Healing potion."
Data "A Tarot card."
Data "An enchanted gauntlet."
Data "Gold coins."
Data "A scroll of summoning."
Data "The key of Ice."
Data "A golden key."
Data "The Great Jem of Goshen."
Data "The magic Sword of Haen."
Data "A small dirk."
Data "The scroll of healing."
Data "The Red Scroll of Fire."
Data "The Blue Scroll of Water."
Data "The Spell of Air."
Data "The mystery Object."
```


All the monsters and opponents have their attributes read into the _OPP_ATTR array. I have two monsters defined. You can re-design them and put your own in, as long as the sequence of types of monster to attribute points is kept the same.

```
'
'... "Skeleton", "Wraith", "Troll", "Ork", "Ghoul", "Mummy"..
' [1] type of opponent [2] strength [3] IQ [4]Hit [5]dexterity [6]magic
 Restore OPP_ATTR_DATA
' set the opponents for encounter 1
'
For T=1 To 6 : For S=1 To 6 : Read OPP_ATTR(T,S) : Next S : Next T
OPP_ATTR_DATA:
Data 1,25,10,20,0,0
Data 2,16,20,30,15,0
Data 0,0,0,0,0,0
Data 0,0,0,0,0,0
Data 0,0,0,0,0,0
Data 0,0,0,0,0,0
End Proc
```

Procedure RIGHT_MOVE

The program line below will take some explaining, as it is fairly important to the control of the character and map movement.

'If MX mod 16=0' will only allow the line to be used if the main character is in the centre of a square.

'TV=Tile Val(etc,' uses a Tome Junior routine to 'look' to the right (+16 pixels) of the character who is positioned on the screen, eighty pixels from the left and ninety-six pixels down from the top of the screen. It then starts him to move his legs and face to the right. As yet there is no movement of the map behind the character.

```
if MX mod 16=0 Then TV=Tile Val(Map Pos X(HX*16+FX+80+16),Map PosY(HY* ↵
16+FY+96-8),0) : Amal 1,RWALK$ : Amal On 1
```

If the variable to the right of the leader is 0 and the leader stands in the centre of a square then TV=0 . The characters Amal flag is turned of so there is no apparent movement and the procedure is left by using Pop Proc. Some programmers object to the use of this command but, if it works well, use it. The design of these four direction routines took a very long time. I could not get the leader through doors or over objects because every time the leader came to a wall or obstruction, the mx, my variable would start to drift.

These few lines fixed that problem.

If TV=0 and MX mod 16=0 Then RV=0 : Amal Off 1 : Pop

MX mod 16 means that if MX was 32 if it were divided by 16 there should be a result of zero. If the variable MX were 34 the result of MX Mod 16 would be 2.

```
.....
If the leader is not in the centre of the square, keep the map scrolling!
If RV>0 Then Inc MX
.....
The centre of a square is reached, But the character still has a way to travel.
If RV<>0 and MX mod 16=0 Then Dec RV
.....
The long treck is over!
If RV=0 and MX mod 16=0 Then Amal Off 1
End Proc
.....
```

The three procedures following are structured the same as the right move with the exception of the addition and subtraction of the value sixteen in the Tile Value (TV) section.

```
Procedure LEFT_MOVE
Procedure _DOWN_MOVE
Procedure UP_MOVE
```

Procedure TSAVE
Saves out the team attributes that are poked into bank fifteen and then saved out to disk as an .ABK file.

```
.....
Procedure TLOAD

' load in the first four team members generated by namegen
'SPECIES=R1
'CLASS=R2
'STRENGTH=R3
'IQ=R4
'HIT=R5
'DEXTERITY=R6
'MAGIC=R7
'ALIGNMENT=R8
'GENDER=R9
'PICTURE NUMBERS
```



```
.....
The names of the team members are used as file names as well.
NAME$(1)=Dir First$("ATTRIBUTES/*.abk")
.....
and scanned into name$() using Dir First$ and Dir Next$
For T=2 To 4
NAME$(T)=Dir Next$
Next T
'
For TEAM=1 To 4 F=Instr(NAME$(TEAM),".") : F=F+3
FILE$=Left$(NAME$(TEAM),F) : Rem strip of trailing spaces
FILE$="ATTRIBUTES/"+Mid$(FILE$,2,Len(FILE$))
NAME$(TEAM)=Left$(NAME$(TEAM),F-4)
Erase 15
Load FILE$,15
.....
```

The attributes in bank fifteen are recovered, nine is subtracted and the result stored in _ATTR_TEAM. The name of the array is probably a bit too long but once the game is finished, the variable names can be shortened with a Search & Replace command.

```
For S=9 To 19
_ATTR(TEAM,S-9)=Peek(Start(15)+S) : Rem  Print _ATTR(TEAM,S-9); : Print " ";
Next S
Next TEAM
TLOAD_PIC
End Proc
.....
Procedure SET_ATTR
Steps through three of the _attr variables and turns the numbers into bars for the panels
CNTR=0
For T=1 To 4
Ink 21
..... STRENGTH .....
Bar 55,5+CNTR To 56+(_ATTR(T,3)/5),8+CNTR Ink 24
..... HIT POINTS .....
Bar 55,15+CNTR To 56+(_ATTR(T,5)/5),18+CNTR Ink 25
..... MAGIC .....
Bar 55,25+CNTR To 56+(_ATTR(T,7)/5),28+CNTR Add CNTR,50
Next T
End Proc
.....
```

Procedure MAPMOUSE
Puts the white square bob on the screen when you click the mouse on the screen.

Procedure ZSET
Sets the zones on the main screen. ZSET is called every time the main game screen is returned to after battle or inventory.

.....

Procedure TEAM1
This routine is called when the mouse has clicked on the top picture on the left hand side of the screen. The variable OLEADER holds the number of the last team member clicked on.

If the character you click on is not dead, '_ATTR(LEADER,0)' then sets the OLEADER to LEADER LEADER=1
:If _ATTR(LEADER,0)=0 Then LEADER=OLEADER

If OLEADER<>LEADER
Checks OLEADER so that you cannot keep clicking on the same person.

WIPETEAM
Turns all the other boxes off.

OLEADER=1
Sets this character to the picture and attributes of LEADER to 1

MAKE_LEADER
Sets all the animation strings for Amal

Ink 24
Gr Writing 2
Box 0,0 To 78,50
Gr Writing 1

Draw a red box around the panel that holds the Statistics and picture of the team member. As the writing style is set to 2 'Gr Writing 2, if the box is re-drawn again over it , the box will vanish. This is what happens in the WIPETEAM procedure.

End If
Wait 5

The Wait 5 is used because the Amiga is so quick that the mouse is read about five or six times in one attempt to click on the correct panel. It works very well and the small delay is not noticed.

End Proc

.....

Procedure TEAM2

Procedure TEAM3

Procedure TEAM4
TEAM2, TEAM3 and TEAM4 are all set out the same as TEAM1

.....

Procedure WIPETEAM
Gr Writing 2
If OLEADER=1 Then Box 0,0 To 78,50
If OLEADER=2 Then Box 0,50 To 78,100
If OLEADER=3 Then Box 0,100 To 78,150
If OLEADER=4 Then Box 0,150 To 78,198
Gr Writing 1

The above procedure draws a box in Gr Writing 2, so that the box 'underneath' vanishes. This style of overwriting to remove the picture underneath could be quite useful for an alert box that did not clear the whole of the screen when it needed to be removed.

End Proc

Chapter 10

- Yet more Procedures
- The conversation maker and Hypertext routine.

```
Procedure SPEAK
.....
If you are still speaking, jump to the Label SPEAK
If SPEAK=True Then Goto Speak
.....
Set the variable speak to true so that the above command is toggled the next time.
SPEAK=True
Get Block 1,0,0,80,200 : Rem get the character panel
.....
Then darken the screen to hide all the putting and getting.
Fade 2 : Wait 20
Bob Off : Bob Update '.....
Screen 1 : Cls 0 : Screen Copy 1 To Logic(0)
Screen Copy 1 To Physic(0)
Screen 1 : Put Block LEADER+6,5,5
' put the pictures in place here
Screen Copy 1,0,0,320,200 To Logic(0),0,0 Screen Copy 1,0,0,320,200 To Physic(0),0,0
Synchro
Screen Swap
Wait Vbl
Screen 0
Reserve Zone 10
.....
Then get the screen colour from the sprite palette.
Fade 1 To -1
.....
No! It isn't a DORK, it's a Dark Ork. I tried to make it an Ork but the Basic interpreted it
as Or K. I suppose I could have called it a Balrog.


If OPP=DWARF Then Restore DWARF_DATA
If OPP=D_ORK Then Restore D_ORK_DATA
For T=1 To 15 : Read MESS$(T) : Next T
.....
```


The square brackets are used to designate a hypertext word. The three questions at the top are all one zone long. The 'answers' give another response if certain words are chosen. I did change the colour of the hypertext word to blue to make finding what word was clickable, but changed it back. It gives more of a challenge to see if a word is hypertext or not. It's up to you.

```
DWARF_DATA:
Data "What is your name and tribe?"
Data "My name is Grimble and I am Hight Son of[Gurgi-Hardpath]. I am chief scout of ↵
the[Kelti]from the caves of[Endili]."
Data "Gurgi-Hardpath is the greatest chief of the Dwarves since Snuri_Ognam drove the ↵
great way between Hendor and the Cave of Steel."
Data "We trace our ancestry from the Briganti. I thought you would know that, Human."
Data "Ah, Endili! How I miss the deep green Pools of Night!"
'
Data "Why are you here?"
Data "I seek the[gold torque]that was stolen by[Orks]from The[Temple-of-Dark]."
Data "It is a great treasure of our people and is reported to have been sighted near here."
Data "Yes! The cursed Orks, spawn of Grimod the Swart."
Data "It is the focus of all Dwarfish lives, a deep and abiding secret, it is most Holy!"
'
Data "Do you come in peace?"
Data "Of course! There is a the[Treaty_of_Halanor]signed in[Electrum]in[Desdsa]."
Data "The treaty was not really needed between our peoples but did ratify our mutual
friendship."
Data "The sacred metal mixture, magical and outlasting all others."
Data "Desda is the Hidden City of the Highest cave in Dwarfdom."
Rem etc. etc.
'..... DIALOGUE LOOP .....
```

The label that is jumped to from the start of the procedure if the routine is already called.

```
Speak:
```

All Labels are ended with a colon after the word. There must be no space between them. This can lead to a problem in the program. If you start a line with a call to a procedure and the colon is placed too close to the procedure call, the basic interpreter thinks the word is a label, and so ignores it.

```
.....
PLOOP1:
QUESTION
If A=10 Then Goto DEPART
PLOOP2:
```

```
ANSWER_TOP_QUESTION
If A=10 Then Goto DEPART
If A<4 Then QUESTION
If A=9 Then Goto PLOOP2
QUESTION_FURTHER
If A<10 Then QUESTION
If A=10 Then Goto DEPART
Goto PLOOP2
.....
'The label DEPART: is used as a jump to label when the variable SPEAK=true
.....
DEPART:
SPEAK=False
' The SPEAK variable toggles on and off and is used when a further question from the top
of the screen is accessed.
.....
The screen is faded and the original game restored
Fade 2,0
' _RESTORE
RE_SCREEN
End Proc
.....
```

The above coding is a kind of tree structure that allows you to go back to each original question and then contine to examine the answers. I tried recursion but got my stack in a twist.

```
'
Procedure QUESTION
'Put Block 3
Reset Zone
Z=1 : Q=False : TY=0 : A=False : TMP=0
Locate 12,1 : Pen 21 : Paper 0 : Print NAME$(LEADER)+" SPEAKS"
TX=13
[ED] one line please
TY=3 : A=True : A$=MESS$(1) : Gosub TIDY : A$=MESS$(6) : Gosub TIDY : A$=
MESS$(11) : Gosub TIDY

Goto FROG
'..... TIDY SUBROUTINE (1) .....
TIDY:
'
R$=A$ : TXT_WIDE=15 : R=TXT_WIDE : OLD_TY=TY
```



```
Repeat
S$=Mid$(R$,R,1)
If S$=" " Then Locate TX,TY : Inc TY : Print Left$(R$,R-1) : R$=Mid$(R$,R+1, (Len
(R$)-R)) : R=TXT_WIDE Else Dec R Until Len(R$)<TXT_WIDE
Locate TX,TY : Print R$ : R$=""
If A=True Then Set Zone Z,TX*8,OLD_TY*8 To 220,TY*8+8 : Inc Z
'If A=True Then Box TX*8,OLD_TY*8 To 220,TY*8+8
Inc TY : Inc TY
Return
```

I often use the above routine in adventure games or to print out a neat report. What I added this time was the zone setting for the hyper-text. Below is an alternative tidy routine for coders with the new, tasty commands from Shadow Software.

```
'..... TIDY SUBROUTINE (2) .....
'
' This tidy routine uses the much quicker command 'wrap' from the Tome Disk
' TIDY:
' L=15 : OLD_TY=TY : TY=TY+10
' While A$<>""
' W=Wrap(A$,L)
' If W>0 and Len(A$)>=5
' B$=Mid$(A$,1,W+1)
' A$=Mid$(A$,W+2)
' Else
' B$=A$
' A$=""
' End If
' Text TX*8-(Text Length(B$)/2),TY,B$
' Add TY,10
' Wend
' If A=True Then Set Zone Z,100,OLD_TY To 220,TY : Inc Z
' y=y-5
'
' Return
FROG:
'set zones for more and exit ( 9 & 10 )
Set Zone 10,260,180 To 300,190 : Text 259,172+16,"EXIT"

Limit Mouse
Fade 1 To -1
Repeat
A=Mouse Zone
```

```
Until Mouse Key=1 and Mouse Zone<>0
TMP=A
End Proc
'
'
Procedure ANSWER_TOP_QUESTION
If A<4 Then Goto BUMP
If TMP=3 Then A$=MESS$(2)
If TMP=8 Then A$=MESS$(7)
If TMP=13 Then A$=MESS$(12)
BUMP:
If A=1 Then A$=MESS$(2) : TMP=3
If A=2 Then A$=MESS$(7) : TMP=8
If A=3 Then A$=MESS$(12) : TMP=13
JUMP:
HYPER[A$]
Limit Mouse
Repeat
A=Mouse Zone
Until Mouse Key=1 and Mouse Zone<>0
End Proc
'
'.....
```

The procedure below read in the words from the array WRD$()

```
Procedure HYPER[R$]
Cls 0,0,110 To 320,170 : Rem clear the text area
Ink 25,0
X=1 : Y=110 : W=1 : PR$="" : For T=4 To 8 : Reset Zone T : Next T : Z=4
R$=R$+"  " : R=38 : S$="" : C$=R$ : DR=0
Repeat
S$=Mid$(R$,R,1)
If(S$=" ") Then WRD$(W)=Left$(R$,R) : Inc W : R$=Mid$(R$,R+1,(Len(R$)-R)) :  ↵
R=38 Else Dec R L=Len(R$)
Until S$="*" or Len(R$)<38
'.....
WRD$(W)=R$ : R$="" : Rem put line of text into wrd$(w) then inc w
For T=0 To W : Rem start scanning wrd$(line)
For S=1 To Len(WRD$(T)) : Rem start scanning the letters in wrd$(line) I$=Mid$  ↵
(WRD$(T),S,1) : Rem the letter is i$
If I$="[" Then I$=" " : Gosub PICKUM : rem you can change the colour of the ink to
highlight the words here.
If I$="]" Then I$=" " : Rem as with the above this is the place to put an ink colour change
```



```
in to set the printing back to it's normal colour.
If I$<>"" Then PR$=PR$+I$
Next S : Rem:drop down to print next line
Text X,Y,PR$ : PR$="" : Y=Y+9
Next T
Rem
Goto PLOOP
'.....

PICKUM:
X1=Text Length(Left$(WRD$(T),S+1)) : Y1=Y-8
For U=S To Len(WRD$(T))
If Mid$(WRD$(T),U,1)="]" Then X2=Text Length(Left$(WRD$(T),U+1)) : Set Zone  ↵
Z,X1,Y1 To X2,Y1+8 : U=Len(WRD$(T)) : Inc Z Next U
Return
PLOOP:
End Proc
Procedure QUESTION_FURTHER
A$=MESS$(TMP+A-4)
HYPER[A$]
Repeat : A=Mouse Zone : Until Mouse Key=1 and Mouse Zone<>0
SPEAK
End Proc
Procedure TAKE
The line below toggles  take between positive and negative each time it is called.
TAKE= Not TAKE
If TAKE=True Then MAIN_SCREEN Else Bell : Wait 5
End Proc
.....
```

I never got to bribe anybody. What I could do with that command is to change it into a buying routine for a weapons shop.

```
Procedure BRIBE
End Proc

Procedure LOOK
.....
```

Just a jolly piece of music from Montiverde. And happily my own arrangement, so that there is no copywrite infringement.

```
EXECUTE["sonix_play > NIL: * i=df1:instruments df1:scores/ritornello"]
AGAIN:
BEGIN_NEW_SCREEN[0] : Rem -1=group
Colour 0,0 : Ink 21,7 : Text 100,10,NAME$(LEADER)+" The "+CLASS_NAME$ ↵
(LEADER)
Fade 1 To -1 : Wait 5
PNTR=30
Reserve Zone 26
```

The lines below look through the leaders array and pastes the object icon on the screen. This way of storing who carries what means that it is possible for all four to carry an object with the same description.

```
For T=1 To 20
If OBJTEAM(LEADER,T)
Paste Icon 85,PNTR-10,(221+T)
Set Zone T,85,PNTR-10 To 101,PNTR+6
```

.....

Here and there through the listing you can see box commands that mirror the zone command. I have found it a useful trick to check that the zones position is where you want it. Just click on the line with the zone command, make a copy with the block command and paste it. All that is needed then is a small change at the start of the line to convert the Set Zone to Box and the zone area is clearly shown when the program is run.

.....

```
' Box 85,PNTR-10 To 101,PNTR+6
Text 102,PNTR,OBJNAME$(T)
PNTR=PNTR+16
End If
Next T
```

.....

```
If OBJTEAM(LEADER,21)<>0 Then Set Zone 21,8,95 To 28,120
If OBJTEAM(LEADER,22)<>0 Then Set Zone 22,30,95 To 50,120
If OBJTEAM(LEADER,21)<>0 Then Paste Icon 10,100,OBJTEAM(LEADER,21)+221
If OBJTEAM(LEADER,22)<>0 Then Paste Icon 32,100,OBJTEAM(LEADER,22)+221
```

The above lines check if the leader is carrying anything and then pastes them below the picture if they are and sets a zone .

.....

SACK variable is used to hold objects carried but not held.
HOLD is used for objects carried in the hand.
USE I haven't used this one yet but it could be used to drink or eat or read.


```
HOLD=False : SACK=False : USE=False
Pen 21 : Locate 2,23 : Print Border$(Zone$("HOLD",23),1)
Locate 8,23 : Print Border$(Zone$("CARRY IN PACK",24),1)
Locate 23,23 : Print Border$(Zone$("USE",25),1)
Repeat
PLOOP:
MZ=Mouse Zone : MK=Mouse Key : If MK=2 Then Goto GOODBYE
If MZ<>0 and MK<>0 Then Wait 5 Else Goto PLOOP
If MZ=23 Then HOLD=True
If MZ=24 Then SACK=True
If MZ=25 Then USE_IT=True

If HOLD and MZ<20 and MZ>0
If OBJTEAM(LEADER,21)=0
OBJTEAM(LEADER,21)=MZ : OBJTEAM(LEADER,MZ)=0 : Paste Icon 10,100, ↵
MZ+221 : Goto AGAIN
End If
If OBJTEAM(LEADER,22)=0
OBJTEAM(LEADER,22)=MZ : OBJTEAM(LEADER,MZ)=0 : Paste Icon 32,100,MZ ↵
+221 : Goto AGAIN
End If
End If
If SACK and MZ=21
M=OBJTEAM(LEADER,21) : OBJTEAM(LEADER,M)=True : OBJTEAM(LEADER, ↵
21)=0 : Goto AGAIN
End If
If SACK and MZ=22
M=OBJTEAM(LEADER,22) : OBJTEAM(LEADER,M)=True : OBJTEAM(LEADER, ↵
22)=0 : Goto AGAIN
End If
If USE_IT and MZ=21 Then USE[OBJTEAM(LEADER,21)] : Goto AGAIN
If USE_IT and MZ=22 Then USE[OBJTEAM(LEADER,22)] : Goto AGAIN Until False
GOODBYE:
LOOK=0 : MZ=0 : MK=0
_RESTORE
End Proc
'
Procedure SWORD
```

.....

```
Just make sure that there are some opponents around.
FLAG=0 : For T=1 To 6 : If OPPONENTS(T)<>0 Then FLAG=T
Next T
If FLAG=0 Then Pop Proc
```

```
.....
EXECUTE["sonix_play > NIL: * i=df1:instruments df1:scores/lady_c"]
Just some suitable battle type music by ANON.
Shared FIGHT,RETREAT,SPELL,PARRY,BOOST,REPORT
PLOOP:
BEGIN_NEW_SCREEN[-1] : Rem (-1) true= group pic
.....
The OOPS variable is used to (hopefully) count down as you defeat the enemy.
OOPS=OPPONENTS(1)+OPPONENTS(2) : Rem add more opponent arrays when you
come to design them.
Bank Swap 2,10 : Rem the opponents pics
Paste Icon 80,170,Length(2) : Rem The fight panel is always loaded in last
Reserve Zone 25
Z=6 : PNTR=1 : MP=1
.....
The enemy array is set up afresh each time there is a battle and the opponents are selected
one by one and placed in it. There are only four enemys per battle, but this method allows
for a different number and range each time there is a confrontation.
.....
'
For T=1 To 6
If OPPONENTS(T)<>0
For S=1 To OPPONENTS(T)
Paste Icon 278,PNTR,T : Inc PNTR : Wait Vbl
ENEMY(MP,1)=T : Rem type of enemy
ENEMY(MP,2)=OPP_ATTR(M,2) : Rem strength ENEMY(MP,3)=OPP_ATTR(M,3) :
Rem IQ ENEMY(MP,4)=OPP_ATTR(M,4) :
Rem HIT ENEMY(MP,5)=OPP_ATTR(M,5) :
Rem DEXTERITY ENEMY(MP,6)=OPP_ATTR(M,6) :
Rem MAGIC Inc MP
Rem Box 278,PNTR To 319,PNTR+50
Set Zone Z,278,PNTR To 319,PNTR+50 : PNTR=PNTR+49 : Inc Z Next S
End If
Next T
'
'.....
Bank Swap 2,10 : Rem and back for the tome tiles
Calls the routine to place objects under the pictures of the team.
STICK_OBJECTS
Cls 0,80,0 To 270,169
Ink 21,7 : Text 80,10,NAME$(LEADER)+" the "+CLASS_NAME$(LEADER) Set Zone ↵
1,0,0 To 78,50
Set Zone 2,0,50 To 78,100
```



```
Set Zone 3,0,100 To 78,150
Set Zone 4,0,150 To 78,198
'
Set Zone 10,80,172 To 132,185 : Rem  ATTACK
' Box 80,172 To 132,185
Set Zone 11,134,172 To 190,185 : Rem RETREAT
' Box 134,172 To 190,185
Set Zone 12,192,172 To 272,185 : Rem CAST SPELL
' Box 192,172 To 272,185
Set Zone 13,80,186 To 132,199 : Rem PARRY
' Box 80,186 To 132,199
Set Zone 14,134,186 To 190,199 : Rem BOOST
' Box 134,186 To 190,199
Set Zone 15,192,186 To 273,199 : Rem REPORT
' Box 192,186 To 273,199
'
Fade 1 To -1
PLOOP2:
I$=Inkey$ : If I$="s" Then STATS
MZ=Mouse Zone : MK=Mouse Key
'
If MK=2 Then Goto GOODBYE
If MK=0 Then Goto PLOOP2
If MZ<5 Then On MZ Proc TEAM1,TEAM2,TEAM3,TEAM4
If MZ>9 and MZ<16 Then WIPE_FIGHT
If MZ=10 Then FIGHT=True
If MZ=11 Then RETREAT=True
If MZ=12 Then SPELL=True
If MZ=13 Then PARRY=True
If MZ=14 Then BOOST=True
If MZ=14 Then REPORT=True
If FIGHT and MZ>4 and MZ<10 Then PUNCHUP[MZ]
If SPELL=True and MZ<10 Then MAGIC[MZ]
If OOPS=0 Then BATTLE=0 : Goto GOODBYE : Rem you have wiped them all out
If OPPONENTS(1)+OPPONENTS(2)<>OOPS Then Repeat : Until Mouse Key<>0 :
Goto PLOOP Goto PLOOP2
GOODBYE:
_RESTORE
'.....MAIN BATTLE LOOP .....
End Proc
'
Procedure DROP[OBJ]
Rem keep obj local
```

```
For T=-1 To 2
For S=-1 To 2
```

.....

You have met this one before in the Procedure RIGHT_MOVE

```
TVL=Tile Val(Map Pos X(HX*16+FX+80+T),Map Pos Y(HY*16+FY+96-16+S),0)
If TVL=128 and OBJ<>0
Poke Start(6)+4+Map Pos X(HX*16+FX+80)+Map Pos Y(HY*16+FY+96-16)*Map ↵
X,OBJ : OBJ=0 : Map Handle Init : Rem if you have the tome extensions delete this line
End If
Next S
Next T
End Proc
Procedure SPELL
Nothing here yet, so put in your own routine
End Proc
```

.....

RESTORE and RE_SCREEN are used to replace the original screen and set up the panels and map

.....

```
Procedure _RESTORE
Fade 1 : Wait 3
Cls 0 : Cls 1
RE_SCREEN
Fade 1 To -1 : Wait 10
ZSET
MK=0 : MZ=0
End Proc
Procedure RE_SCREEN
HX=Map Hx(MX)
HY=Map Hy(MY)
FX=Map Fx(MX)
FY=Map Fy(MY)
Screen 1
Map Handle 1,HX,HY
Map Do HX,HY
Screen 0
Put Block 1 : Put Block 2 : SET_ATTR


For T=1 To 4
If OBJTEAM(T,21)<>0 Then Paste Icon 43,T*50-18,OBJTEAM(T,21)+221
If OBJTEAM(T,22)<>0 Then Paste Icon 59,T*50-18,OBJTEAM(T,22)+221
```



```
Next T
Screen Copy Logic To Physic
Synchro
Screen Copy 1,FX,FY,240+FX,172+FY To Logic(0),80,0
MAKE_LEADER
Bob 1,32*5+6,32*3-4,T+13
Bob 2,-200,-200,6
Bob Draw
Screen Swap
Wait Vbl
End Proc
Procedure MAIN_SCREEN
If TAKE Then GRABBIT : Goto FROG
```

.....

TAKE tells the program that to jump to GRABBIT even though the action takes place on the Map area.

```
'
X=X Screen(X Mouse)-80
Y=Y Screen(Y Mouse)
TVL=Tile Val((MX+X)/16,(MY+Y)/16,0)
MPOSX=X Screen(X Mouse)/16*16
MPOSY=Y Screen(Y Mouse)/16*16+16
 Bob 15,X1,Y1,1 : Amal On 15
```

.....

Sets the values that are used by the Procedures RIGHT_MOVE LEFT_MOVE,etc.

```
'..... DOWN .....
 If MPOSY>97 Then DV=MPOSY-96 : DV=DV/16 :
'..... UP .....
If MPOSY<95 Then UV=MPOSY-97 : UV=Abs(UV) : UV=UV/16
 '..... RIGHT .....
If MPOSX>161 Then RV=MPOSX-160 : RV=RV/16
'..... LEFT .....
If MPOSX<159 Then LV=MPOSX-160 : LV=Abs(LV) : LV=LV/16
FROG:
End Proc

Procedure TLOAD_PIC
Erase 2
Load "team/wreath_icon.abk"
```

.....

In the lines of code below, the ,2 behind the bank is important. This loads the team pictures in behind the wreath. Without the comma it would load and remove anything that was there already.

```
For TEAM=1 To 4
T=_ATTR(TEAM,10) : T$=Str$(T) : Rem get past the wreath, ( icon number 1)/ ( square
mouse pointer bob)
Load "TEAM/ICON"+T$+".abk",2 : Rem concatination
Load "TEAM/BOB"+T$+".abk",1 : Rem more concatination
Next TEAM
Make Icon Mask
Load "OPPONENTS/icon1.abk",2 : Load "OPPONENTS/icon2.abk",2 : Rem load in the
small pictures of opponents
Load "OPPONENTS/bob1.abk",1 : Load "OPPONENTS/bob2.abk",1 : Rem load in bob
anims of opponents
End Proc

Procedure MAKE_LEADER
If LEADER=1 Then T=5
If LEADER=2 Then T=23
If LEADER=3 Then T=41
If LEADER=4 Then T=59


LWALK$=" A 0,("+Str$(1+T)+",3)("+Str$(2+T)+",3)("+Str$(3+T)+",3)("+Str$(4+T)+
",3)("+Str$(5+T)+",3)("+Str$(6+T)+",3) "
RWALK$=" A 0,("+Str$($8000+1+T)+",3)("+Str$($8000+2+T)+",3)("+Str$($8000+3+
T)+",3)("+Str$($8000+4+T)+",3)("+Str$($8000+5+T)+",3)("+Str$($8000+6+T)+",3) "
UWALK$=" A 0,("+Str$(T+7)+",3)("+Str$(8+T)+",3)("+Str$(9+T)+",3)("+Str$(10+T)
+",3)("+Str$(11+T)+",3)("+Str$(12+T)+",3)"
DWALK$=" A 0,("+Str$(13+T)+",3)("+Str$(14+T)+",3)("+Str$(215+T)+",3)("+Str$
(16+T)+",3)("+Str$(17+T)+",3)("+Str$(18+T)+",3)"

Bob 1,,,T+13
End Proc

Procedure MAKE_OPPONENT

If OPPONENT=1 Then OPP=1
If OPPONENT=2 Then OPP=13
RGMARCH$=" A 0,("+Str$(OPP+77)+",4)("+Str$(OPP+78)+",4)("+Str$(OPP+79)+"
,4)("+Str$(OPP+80)+",4);L : L X=RA;L Y=RB;P;J L "
LGMARCH$=" A 0,("+Str$($8000+77+OPP)+",4)("+Str$($8000+78+OPP)+",4)("+Str
$($8000+79+OPP)+",4)("+Str$($8000+80+OPP)+",4) ;L: L X=RA ; L Y=RB ; P ; J L "
End Proc
```



```
Procedure SKELETON
GTIM=GTIM+GM
If GTIM>80
GM=-1 : Amal 2,LGMARCH$ : Amal On 2
End If
If GTIM<4
GM=1 : Amal 2,RGMARCH$
Amal On 2
End If
Amreg(0)=650-((HX*16+FX)+GTIM)
Amreg(1)=(200+16)-(HY*16+FY)
End Proc

Procedure WRAITH
GTIM=GTIM+GM
If GTIM>80
GM=-1 : Amal 2,LGMARCH$ : Amal On 2
End If

If GTIM<4
GM=1 : Amal 2,RGMARCH$ : Amal On 2
End If
```

The Amreg [] send the correct co-ordinates to the monsters and opponents so that they do not drift around the map. Each seperate opponent needs a different set of Amregs.

```
Amreg(0)=650-((HX*16+FX)+GTIM)
Amreg(1)=(200+16)-(HY*16+FY)
End Proc

Procedure TRAP
Just checking!
Boom
End Proc

Procedure BEGIN_NEW_SCREEN[T]
Fade 2 : Wait 10
Bob Off : Bob Update '.....
Screen 1 : Cls 0 : Screen Copy 1 To Logic(0) : Screen Copy 1 To Physic(0) Screen 1 : If
T=0 Then Put Block LEADER+6,5,5 Else Put Block 1
SET_ATTR Screen Copy 1,0,0,320,200 To Logic(0),0,0
Screen Copy 1,0,0,320,200 To Physic(0),0,0 Synchro
Screen Swap
```

```
Wait Vbl
Screen 0
End Proc


Procedure GRABBIT

TVL=Tile Val(Map Pos X(HX*16+FX+80),Map Pos Y(HY*16+FY+96-16),0)
```

If the number of the tile the character is standing on is over zero and under fifty then it is an object that can be picked up.

```
If TVL>0 and TVL<50
OBJTEAM(LEADER,TVL)=True
```

```
Poke Start(6)+4+Map Pos X(HX*16+FX+80)+Map Pos Y(HY*16+FY+96-16)*Map ↵
X,196 : Map Handle Init
```

This pokes into the map memory the number of a plain floor tile, and then calls the Hand Init to put it on the screen. If you have the Tome extensions this can be handled much more efficiently.

```
End If
TAKE=False
DV=0 : LV=0 : RV=0 : UV=0
End Proc

Procedure TIDY[R$]
Cls 0,80,0 To 270,169
TXT_WIDE=23 : R=TXT_WIDE : TX=10 : TY=2 : OLD_TY=TY
Repeat
S$=Mid$(R$,R,1)
If S$=" " Then Locate TX,TY : Inc TY : Print Left$(R$,R-1) : R$=Mid$(R$,R+1, (Len ↵
(R$)-R)) : R=TXT_WIDE Else Dec R Until Len(R$)<TXT_WIDE
Locate TX,TY : Print R$ : R$=""
Inc TY : Inc TY
End Proc

Procedure USE[M]
```

Use is not used yet.

```
Shoot : Locate 0,22 : Print M
End Proc
```



```
Procedure STICK_OBJECTS
```

Looks at who is carrying what and sticks the object's icon below the picture of the team member in the panel.

```
For T=1 To 4
If OBJTEAM(T,21)<>0 Then Paste Icon 43,T*50-18,OBJTEAM(T,21)+221
If OBJTEAM(T,22)<>0 Then Paste Icon 59,T*50-18,OBJTEAM(T,22)+221
Next T
End Proc

Procedure PUNCHUP[M]
Pen 21 : Paper 7
M=M-5
 L=LEADER : Rem I got fed up typing in the word leader.
```

if the red potion is carried

```
If OBJTEAM(L,21)=4 or OBJTEAM(L,22)=4
```

```
R$=R$+NAME$(L)+" drinks from the red bottle and with the extra power from the ↵
potion "
If OBJTEAM(L,21)=4
OBJTEAM(L,21)=0
Add _ATTR(L,3),10 : Rem gains extra power to attack
Dec _ATTR(L,6) : Rem but he becomes clumsy
Else OBJTEAM(L,22)=0
End If
End If
R$=R$+NAME$(L)+" attacks a "+OPPNAME$(ENEMY(M,1)) : Rem enemy holds the ↵
type of monster
'

If OBJTEAM(L,21)<>0 Then R$=R$+" with "+OBJNAME$(OBJTEAM(L,21))
If OBJTEAM(L,22)<>0 Then R$=R$+" and  "+OBJNAME$(OBJTEAM(L,22))
If _ATTR(L,3)<ENEMY(M,2)
R$=R$+" but the strike has little effect!" : ENEMY(M,2)=ENEMY(M,2)-2 : Dec _ATTR ↵
(L,3)
End If
If _ATTR(L,3)>ENEMY(M,2)
R$=R$+" and hits it "
ENEMY(M,2)=ENEMY(M,2)-5 : Inc _ATTR(L,5)
End If
If _ATTR(L,3)>(ENEMY(M,2)*2)
```

```
Dec OPPONENTS(ENEMY(M,1)) DROP[Rnd(19)+1]
ENEMY(M,1)=0 : R$=R$+" and smashes it"
TIDY[R$]
Pop Proc
End If
```

What the team member has done is printed to the screen

```
TIDY[R$]
```

And then wait to read it until the mouse key is pressed. I tried putting a pause in but it never seemed to last the right length of time. There is nothing worse than having a long interesting fight report cut off in mid punch-up.

```
Repeat : Until Mouse Key<>0

'..... The enemys Turn.....
R$="" : R$="The "+OPPNAME$(ENEMY(M,1))

If ENEMY(M,1)=1 : Rem is a skeleton
'
' a cowardly ( or prudent ) skeleton!
If ENEMY(M,2)<3 and Rnd(10)<7
R$=R$+" turns and lurches off into the darkness." ENEMY(M,1)=0 : TIDY[R$] : Pop ↵
Proc
End If
End If
'
```

If a skeleton attacks a Made, even a new one with low Magic powers, he is asking for trouble!

```
If ENEMY(M,2)<_ATTR(L,3) and _ATTR(L,2)=4
R$=R$+" and is repelled by a flash of Madge Lightning!" TIDY[R$] : Pop Proc
End If
'
The Skeleton attacks!
If ENEMY(M,1)=1
If 25<(Rnd(100))
R$=R$+" grabs and slashes with a bony claw! "+NAME$(L)
Else
R$=R$+" jumps forward and slashes with a sword! "+NAME$(L)
End If
End If
'
```



```
If the opponent is a Wraith.
If ENEMY(M,1)=2
'
The wraith attacks!
If 25<(Rnd(100))
R$=R$+" attacks with a small dagger! "+NAME$(L)
Else
R$=R$+" jumps forward and slashes with her long green nails! "+NAME$(L)
End If
End If
'
If 25>(Rnd(100)) and _ATTR(L,6)>ENEMY(M,5) Then R$=R$+"leaps back and avoids ↵
the attack." : TIDY[R$] : Pop Proc
If 25>(Rnd(100)) Then R$=R$+" blocks the attack." : TIDY[R$] : Pop Proc
R=Rnd(100)
If R<25
R$=R$+" is injured on the arm." : Dec _ATTR(L,3) : Dec _ATTR(1,5) : TIDY[R$] : Pop ↵
Proc
End If
If R>75
R$=R$+" is cut on the leg." : Dec _ATTR(L,3) : _ATTR(1,5)=_ATTR(L,5)-3 : TIDY ↵
[R$] : Pop Proc
End If
If R=>25 and R<=75
R$=R$+" is injured on the chest." : Dec _ATTR(L,3) : Dec _ATTR(1,5) : TIDY[R$] : ↵
Pop Proc
End If
TIDY[R$]
End Proc
```

Procedure STATS

This procedure is not really for the people playing the game but for the game designer. You will have to keep accessing the stats to see how the game is going on. This procedure does that and can be deleted at the end of the game or left in if you think the stats will be of use in the game.

```
Print At(11,1);NAME$(LEADER)
Print At(11,2);"SPECIES   "; : Print SPECIES_NAME$(_ATTR(LEADER,1))
Print At(11,3);"CLASS     "; : Print CLASS_NAME$(_ATTR(LEADER,2))
Print At(11,4);"STRENGTH  "; : Print _ATTR(LEADER,3)
Print At(11,5);"IQ        "; : Print _ATTR(LEADER,4)
Print At(11,6);"HIT       "; : Print _ATTR(LEADER,5)
Print At(11,7);"DEXTERITY "; : Print _ATTR(LEADER,6)
```

```
Print At(11,8);"MAGIC     "; : Print _ATTR(LEADER,7)
Print At(11,9);"EXPERIENCE"; : Print _ATTR(LEADER,8)
Print At(11,10);"GENDER   "; : Print _ATTR(LEADER,9)
End Proc
```

Procedure WIPE_FIGHT
Small but useful, clears all the variables for the fight sequence.

```
FIGHT=False : RETREAT=False : SPELL=False : PARRY=False : BOOST=False :
REPORT=False
End Proc
```

Procedure MAGIC[M]
One of the few sound effect procedures in the game. I like it so much, I might put a few more in.

```
Locate 0,0 : Print MZ
Wave 1 To %1111
For Q=1 To 8
T=Rnd(15)
V=Rnd(15)
Play 1,96-T,3
Next Q
End Proc
```

Procedure EXECUTE[COMMAND$]
Calls a AMIGA command from inside Amos. It is used by the SONIX_PLAY module in the C library to play tunes.

```
TEMP$=String$(" ",1024)
COMMAND$=COMMAND$+Chr$(0)
Dreg(1)=Varptr(COMMAND$)
Dreg(2)=0
Dreg(3)=0
D=Doscall(-222)
End Proc
```

Procedure GO_BATTLE
Sorts out how many opponts are going to fight you

BATTLE is a variable that tells the computer to progress to the next battle scenario.

```
If BATTLE=1
OPPONENTS(1)=2 : SKELETON : Rem there are 2 skeletons OPPONENTS(2)=1 : Rem
there is one wraith
```



```
BATTLE=0
End If
If BATTLE=2
OPPONENTS(1)=1
OPPONENTS(2)=2
BATTLE=0
End If
End Proc
```

Chapter 11

- Selling the Game.
- Fronts and backs.
- Debugging games.
- Virus checkers.

If you want to write games for yourself or a few friends to play with, then you still need to approach the game as though it was to be published by a Big Company, well, OK then, a Budget Company. There is a real buzz to be had when a decent loading screen appears and there is a blast of music, heralding the start of the game. In some commercial games that is the best part. The rest may fall flat on its face with bad game-play and pictures by Naff Graffix Inc. Blank, blunt endings are a real let down too.

If your poor game player has taken weeks or months to battle monsters and deadly traps and survive your game, they certainly deserve a decent end screen and win sequence and not a short, GAME OVER or THE END. Try to make it worth their while. If you think your game was that good, there is nothing to stop you doing a subtle advert for your next block busting extravaganza.

How about 'The team rest weary from their battle, yet Victorious. Soon they will rise and do battle with "SON OF THE SLIME PART II"'. I know it's a bit O.T.T but you get the idea. As the game is finished, there is nothing to be lost by loading in a completely new set of graphics, palette colours, music and animations. Chances are there will be a few spare bits of scenery left over from the game and an occasional font looking for a good home. Try to make the ending memorable.

However there is a chance that you would like to see your game on the shelves and have a boost to your pocket money, Social Security or whatever. I can't say that following this book will do that for you, but there are always new things to learn and friends that can advise and help. Once your game is finished and as polished as you can get it, try it out on your friends. A simple game with good game-play and simple clean graphics will win hands down over a hacked game idea and sloppy program. Look at Tetris. Simple and brilliant!

Bugged games are the worst. If a player boots up the computer which then crashes or grinds to a halt half way through, your programming 'cred' will quickly fade to nothing. Games should be tested, if possible on as many Amiga machines with different memory requirements. Friends can help you by playing the game with you watching both them and the problems they encounter. It can be both instructive and dispiriting to stand behind someone playing your game for the first time. You can usually discover most changes and alterations that are needed in the first few minutes. Others emerge weeks later and can cause many sleepless nights.

Instant death is another annoying feature in a Dungeon and Damp Bits Game! At least give your team a fighting chance against what is going to destroy them. If the danger is too great for them, then let them have the opportunity to retreat and find another way around the obstacle or muster their forces. If the game is too difficult to start with, the player will soon lose interest and your game will end up in the dustbin or re-formatted.

There is also the problem of making the game too easy. If your adventurers breeze through the game with little opposition, there will be no satisfaction when the game is finished. It might be possible to program into the game a semi-intelligent 'spy'. This piece of programming would keep an eye on the team's progress in the game.

In pseudo code this will look like...

If team is doing too well in x number of turns then
Bring on more nasties.
If team is about right then
Do nothing.
If team is getting nowhere in x number of turns then
Give team a hint.
If team is turning out to be a right bunch of raspberries then
Remove all nasties and be extra nice to them.
If team is totally scuppered, the fault is probably your fault for making the game-play too difficult!

If the game does not progress satisfactorily in a set number of turns, drop hints and tips on how to get around the obstacle or solve the problem.

The most cringy thing that can happen is that after the game has been played and tested on a friends computer, the computer crashes. You find that you have introduced a Virus into the friends system. Nasty. Always have a recent and approved virus tester resident on Hard Drive if you have one, or site a floppy disk with a Virus checker next to your trusty Amiga. If a games tester at a company finds that a virus on your cherished game has crashed and mangled his hard drive, guess what your chances are at getting it published?

You may have heard that the Big Companies are unlikely to publish a game submitted in AMOS. Well, unfortunately, it's true. However, by not telling them that it's an AMOS program, there is a good chance that if you keep quiet about the programs source and the program is compiled and the obvious 'A' symbol is removed from the workbench icon, it will pass. Incidentally, if you have an A1200, changing the icons and drawers is a doddle with Icon Edit.

Europress, who publish AMOS, do ask that you inform them about it as soon as the program is accepted by the publishers. Not before! It's not fair to raise their hopes too high as well as your own. If the Publishing company accept your code and then makes a lot of money out of your game, chances are they won't be too put out when they discover the program is AMOS based. Personally I'm biased, I don't see that AMOS based games are any worse (or better) than a lot of the games out now.

Europress's address is:

Europress Software,
Europa House,
Addlington Park,
Macclesfield,
Cheshire.
SK10-4NP

If you are worried that the company may steal your ideas or graphics, post a complete copy of the game to yourself in a registered letter or package. Do not open it and keep it in a very safe place. This is your proof that the game was designed by you. If problems do arise and your hacked game does appear, take the UNOPENED package with your game in to a competent lawyer or seek Legal Aid. There are a lot of Sharks out their as well as Pirates!

Well that's nearly it. There are some bits that can be revised and there are some bits which probably will be, as soon as I send this document off to Kuma. The opponent routine for the Amal could be streamlined and tightened up and extra graphics here and there revised. But then, that's the beauty of Amos. It can be tinkered with and revised as you go along.

I must say thanks to them all at Kuma for their patience and kind advice. Thanks also to Len Tucker and Aaron for their problem solving and knowledge of publishing books. Thanks also to Brian for encouragement and coffee and Glyn Bach (The Dog) for dragging me out for walks.

I think that's all for now. Is it? UM...well there is the TAKE procedure to tighten up and the MAGIC routine to re-design and.....

Bibliography

Here, at the very end is the bibliography. All of these books were a source of inspiration and a great pool in which to dip when the ideas ran dry. As I said earlier, the library is the best place of all to find new sources and references for stories.

The Animators Workbook.
By Tony White.
Published by Phaidon Press
ISBN 0 7148 2439 9 (Hbk)
ISBN 0 7148 2566 2 (Pbk)
Very, very useful for the really puzzling parts of animation. It makes it seem so easy.

Piranesi.
By Nicholas Penny.
Published by Bloomsbury Books, London.
ISBN 1 870630 50 5
Moody and architectural. Great dungeons from the master dungeon builder.

Magic Symbols.
By Pearl Binder.
Published by Hamlyn.
ISBN 0 600 02545 4
If you must use magic spells and talismans, this book will help in the design.

The Silver Arm.
By Jim Fitzpatric.
Published by Dragons World.
ISBN 0 905895 54 1
A feast for the eyes and a great source book for Celtic Stories.

The Land of Froud.
By Brian Froud.
Published by Pan.
ISBN 0 330 25350 6
Goblins, faries and trolls, all beautifuly drawn.

Writing Role Playing Games in AMOS

by Dicon Peeke

Classic Amos Basic is the immensely popular programming system for the whole range of Commodore Amiga's, ideal for a wide variety of applications from games to educational programs.

This book gives ideas, tips and inside information on writing Role Playing Adventures.

The book uses the Role Playing game "The Black Tarot" that comes on the FREE disk (that can be claimed using the coupon included in this book) to explain programming and procedures.

Also included on the disk is a copy of the New Amazing Tome Junior. This fully working Amiga program allows the games-writer to generate huge maps using very little memory using the free extra commands that can be added to your AMOS Library.

There are screens of graphics for you to cut out and use, tile banks, music, icons and bobs. There is a program that tells a different story each time it is run and a program that generates different characters for you to load into the game. All of the programs are fully listed with comments, enabling you to get started programming easily and quickly.

Tim Moore, Managing Director of Kuma said, "This book will help Amiga owners get the best from Amos. A very useful contribution from Dicon".

Published by

Kuma

Kuma Books Ltd.
Pangbourne, Berkshire, England
Tel: 0734 - 844335
Fax: 0734 - 844339

ISBN 0-7457-0247-3

01295

9 780745 702476

£12.95 net